2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Dynamic Locomotion and Whole-Body Control for Quadrupedal Robots

C. Dario Bellicoso, Fabian Jenelten, Péter Fankhauser, Christian Gehring, Jemin Hwangbo, Marco Hutter

Abstract— **This paper presents a framework which allows a quadrupedal robot to execute dynamic gaits including trot, pace and dynamic lateral walk, as well as a smooth transition between them. Our method relies on an online ZMP based motion planner which continuously updates the reference motion trajectory as a function of the contact schedule and the state of the robot. The planner is coupled with a hierarchical whole-body controller which optimizes the whole-body motion and contact forces by solving a cascade of prioritized tasks. We tested our framework on ANYmal, a fully torque controllable quadrupedal robot which is actuated by series-elastic actuators.**

I. Introduction

Research focused on robotic locomotion has made great efforts in trying to reach the natural capabilities of animals, which are capable of executing highly dynamic gaits and naturally perform quick and smooth transitions from one gait to another. The recent advances in both the motion planning and control methods, as well as the increase in performance of both hardware and software, are narrowing the gap between legged robots and their biological counterparts. Although we are still far from a fully autonomous system capable of locomotion in all environments, the recent years have seen several important results.

Dynamic locomotion is a complex problem which involves a multi-body system which interacts with the environment through several contact points. The execution of dynamic gaits for such a system over unknown and challenging terrain requires careful motion planning and accurate motion and force control. Research groups and companies have demonstrated the ability to walk, run and jump over challenging terrain. *Cheetah 2* [15] from MIT has been shown able to run and jump over obstacles which are 80% of the robot's leg length using an MPC controller. Recent results have shown *ANYmal* [9] walking [1], trotting [5], and climbing over stairs [3] by employing whole-body control and motion optimization algorithms. The IIT *HyQ* [17] quadruped has been shown walking using a creeping gait [14] over stepping stones [19]. Motion planning for the torso was obtained by solving an optimization problem which constrained the *Zero-Moment Point* (ZMP) [18] to lie in the support polygons.

A notable example of dynamic locomotion is given by the legged robots developed by Boston Dynamics. One of their recent legged robots, *Spot*, has demonstrated a wide variety of dynamic gaits, including a dynamic walk, trotting with full flight phases, pacing, as well as demonstrating smooth transitions between these. *Spot mini*, a more recent machine, has shown similar skills as well as manipulation capability with a robotic arm. While impressive, the methods to achieve such results are not available. In this paper, we contribute a novel framework and methods for motion planning and control algorithms needed to execute complex and dynamic gaits.

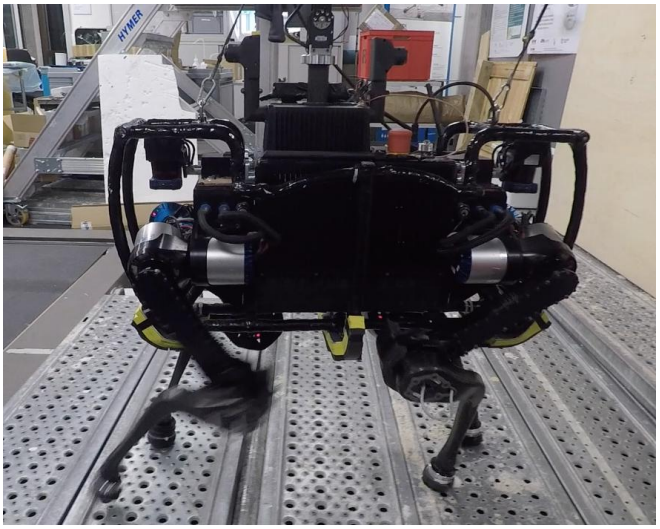

Fig. 1. Our motion plan and control framework, tested on the fully torque-controllable quadrupedal robot ANYmal, can execute a dynamic walk which includes overlapping swing phases for lateral legs (e.g. left-front and left-hind legs).

Based on the results shown in [10] and [19], we developed a ZMP based motion planner which can generate a motion through a sequence of arbitrary support polygons, including triangles and quadrilaterals, but also lines, which is the case for a dynamic walk with overlapping swing phases or for a trotting and pacing gait. Rather than computing the optimal motion on discrete events (e.g. a foot touchdown), we recompute the optimization as soon as the previous one ends successfully, providing us with a motion plan which continuously adapts to the new state of the robot as well as to changes in the gait pattern. We also reformulate the cost function to include costs on the deviation from previous solutions, as well as costs which penalize the deviation from a path regularizer. Motion tracking is obtained by means of a whole-body controller which solves a sequence of motion and contact force prioritized tasks using the hieararchical optimization formulation. We demonstrate our method by executing dynamic gaits such as a dynamic lateral walk with overlapping swing phases, a trot and a pace on the

This work was supported in part by the Swiss National Science Foundation (SNF) through the National Centre of Competence in Research Robotics.

All authors are with the Robotic Systems Lab, ETH Zurich, Switzerland, bellicoso@mavt.ethz.ch

quadrupedal robot ANYmal. We conclude our experiments by showing a transition between a dynamic lateral walk and a pace gait. To the best of our knowledge, this is the first contribution with results and methodologies on the execution of such dynamic gaits, specifically a dynamic lateral walk, a pacing gait and transitions between them.

II. Model Formulation

In general, a walking robot can be modeled as a system with a free-floating base B to which legs are attached (see Fig.2). The motion of the entire system can be described w.r.t. a fixed inertial frame I. We write the position vector from the inertial to the base frame expressed in the inertial frame as ${}_I\mathbf{r}_{IB} \in \mathbb{R}^3$ and use Hamiltonian unit quaternions to parametrize the orientation of the main body. The limb joint angles are stacked in the vector $\mathbf{q}_j \in \mathbb{R}^{n_j}$. We write the generalized coordinates vector $\mathbf{q}$ and the generalized velocities vector $\mathbf{u}$ as

$$\mathbf{q} = \begin{bmatrix} {}_I\mathbf{r}_{IB} \\ \mathbf{q}_{IB} \\ \mathbf{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j}, \quad \mathbf{u} = \begin{bmatrix} {}_I\mathbf{v}_B \\ {}_B\boldsymbol{\omega}_{IB} \\ \dot{\mathbf{q}}_j \end{bmatrix} \in \mathbb{R}^{n_u}, \quad (1)$$

where $n_u = 6 + n_j$, ${}_I\mathbf{v}_B$ and ${}_B\boldsymbol{\omega}_{IB}$ are the linear and angular velocity of the base w.r.t. the inertial frame expressed respectively in the I and B frame, and $\mathbf{q}_{IB} \in SO(3)$ is the unit quaternion that projects the components of a vector expressed in B frame to those of the same vector expressed in I frame. The equations of motion of legged systems can be written as $\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{h}(\mathbf{q}, \mathbf{u}) = \mathbf{S}^T\boldsymbol{\tau} + \mathbf{J}_s^T\boldsymbol{\lambda}$, where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n_u \times n_u}$ is the mass matrix and $\mathbf{h}(\mathbf{q}, \mathbf{u}) \in \mathbb{R}^{n_u}$ is the vector of Coriolis, centrifugal and gravity terms. The selection matrix $\mathbf{S} = \begin{bmatrix} \mathbf{0}_{n_\tau \times (n_u - n_\tau)} & \mathbb{I}_{n_\tau \times n_\tau} \end{bmatrix}$ selects which DoFs are actuated. If all limb joints are actuated, then $n_\tau = n_j$. The vector of constraint forces $\boldsymbol{\lambda}$ is mapped to the joint space torques through the support Jacobian $\mathbf{J}_s \in \mathbb{R}^{3n_c \times n_u}$, which is obtained by stacking the constraint Jacobians as $\mathbf{J}_s = \begin{bmatrix} \mathbf{J}_{C_1}^T & \cdots & \mathbf{J}_{C_{n_c}}^T \end{bmatrix}^T$, with n_c the number of limbs in contact. Motion at the supporting contact points must be constrained. If the feet are modeled as point contacts, then each contact introduces three constraint equations ${}_I\mathbf{r}_{IC}(t) = const$, which can be differentiated twice to yield

$${}_I\dot{\mathbf{r}}_{IC} = \mathbf{J}_C\mathbf{u} = \mathbf{0}, \quad {}_I\ddot{\mathbf{r}}_{IC} = \mathbf{J}_C\dot{\mathbf{u}} + \dot{\mathbf{J}}_C\mathbf{u} = \mathbf{0}. \quad (2)$$

III. Hierarchical Optimization

We have recently shown [1] the implementation of a whole body controller which finds the optimal joint torques and joint accelerations by solving a cascade of prioritized QP problems. In this work we use a different implementation which solves for contact forces rather than joint torques. Hence, we search for a vector $\boldsymbol{\xi}_d = \begin{bmatrix} \dot{\mathbf{u}}_d^T & \boldsymbol{\lambda}_d^T \end{bmatrix}^T \in \mathbb{R}^{n_u + n_c}$ of desired joint accelerations and contact forces. The dimension of ξ_d is a function of the number of contact points n_c. When the feet are modeled as point contacts, the dimensions of ξ_d will be smaller or equal to the case of optimizing for joint torques. As noted in [8], this is beneficial in terms of computation speed when solving the numerical optimization

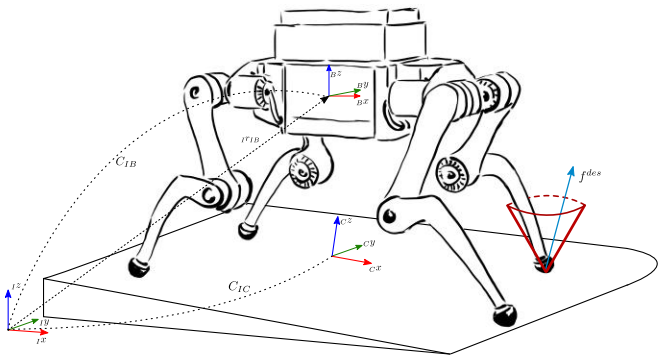

Fig. 2. A sketch of the quadrupedal robot ANYmal. The floating base pose is defined by the position of the base frame wr.t. the inertial frame expressed in the inertial frame ${}_I\mathbf{r}_{IB}$, and by the orientation of the base w.r.t. the inertial frame $\mathbf{C}_{IB}$. As discussed in earlier works [5], we define a control frame $\mathbf{C}_{IC}$ which is aligned with the local estimation of the terrain and with the robot's heading direction.

of tasks. It also results in an more natural way of expressing constraints of contact forces.

Equations of motion: By taking advantage of the decomposition induced by the selection matrix $\mathbf{S}^T$, we can constrain the motion and the contact forces to live on the manifold described by the floating base system dynamics by writing

$$\begin{bmatrix} \mathbf{M}_{fb} & -\mathbf{J}_{s_{fb}}^T \end{bmatrix} \boldsymbol{\xi}_d = -\mathbf{h}_{fb}, \quad (3)$$

where $\mathbf{M}_{fb}$, $\mathbf{J}_{s_{fb}}^T$ and $\mathbf{h}_{fb}$ are the first six rows of the composite inertia matrix, the support Jacobian and the nonlinear terms respectively, which are the equations related to the dynamics of the floating base.

Contact motion constraint: The solution found by the controller should not violate the contact constraints defined in (2). Hence, we impose null accelerations at the contact points by setting

$$\begin{bmatrix} \mathbf{J}_s & \mathbf{0}_{3n_c \times 3n_c} \end{bmatrix} \boldsymbol{\xi}_d = -\dot{\mathbf{J}}_s\mathbf{u}. \quad (4)$$

Contact force and torque limits: To avoid slipping, contact forces should be constrained to lie in the friction cone which is aligned with the normal to the contact surface ${}_I\mathbf{n}$ expressed in world frame and is a function of the friction coefficient μ. To write these constraints as linear ones, we approximate the friction cones with pyramids. These are aligned with the unit perpendicular vectors ${}_I\mathbf{l}$ and ${}_I\mathbf{h}$ which lie on the contact surface. We achieve this by writing constraints on the k-th contact force $\boldsymbol{\lambda}_k$ as

$$\begin{aligned} ({}_I\mathbf{h} - {}_I\mathbf{n}\mu)^T {}_I\boldsymbol{\lambda}_k \leq \mathbf{0} \\ -({}_I\mathbf{h} + {}_I\mathbf{n}\mu)^T {}_I\boldsymbol{\lambda}_k \leq \mathbf{0} \\ ({}_I\mathbf{l} - {}_I\mathbf{n}\mu)^T {}_I\boldsymbol{\lambda}_k \leq \mathbf{0} \\ -({}_I\mathbf{l} + {}_I\mathbf{n}\mu)^T {}_I\boldsymbol{\lambda}_k \leq \mathbf{0}. \end{aligned} \quad (5)$$

Joint actuation torques should also be limited to avoid mechanical limits given by the actuators. We can write

$$\boldsymbol{\tau}_{min} - \mathbf{h}_j \leq \begin{bmatrix} \mathbf{M}_j & -\mathbf{J}_{s_j}^T \end{bmatrix} \boldsymbol{\xi}_d \leq \boldsymbol{\tau}_{max} - \mathbf{h}_j \quad , \quad (6)$$

TABLE I

THE LIST OF PRIORITIZED TASKS USED IN OUR EXPERIMENTS. EACH TASK IS ASSOCIATED WITH A PRIORITY (1 IS THE HIGHEST).

Priority	Task
1	Floating base equations of motion
2	Torque limits
	Friction cone and $\boldsymbol{\lambda}$ modulation
3	No contact motion
4	Center of mass linear motion
	Center of mass angular motion
	Swing leg motion tracking
5	Contact force minimization

where the j subscript is relative to the rows of the equations of motion describing the joint dynamics. The quantities $\boldsymbol{\tau}_{min}$ and $\boldsymbol{\tau}_{max}$ are n_j dimensional vectors of minimum and maximum actuator torques.

Motion tracking: To track the desired motion of the floating base and the swing legs, we constrain the joint accelerations by implementing operational space controllers with feed-forward reference acceleration and a motion dependent state feedback state. For the main body we write

$$\begin{bmatrix} {}_C\mathbf{J}_P & \mathbf{0} \end{bmatrix} \boldsymbol{\xi}_d = {}_C\ddot{\mathbf{r}}^d_{IB} + \mathbf{k}^{pos}_D({}_C\dot{\mathbf{r}}^d_{IB} - {}_C\mathbf{v}_B) + \mathbf{k}^{pos}_P({}_C\mathbf{r}^d_{IB} - {}_C\mathbf{r}_{IB}) \tag{7}$$

for the linear motion and

$$\begin{bmatrix} {}_C\mathbf{J}_R & \mathbf{0} \end{bmatrix} \boldsymbol{\xi}_d = -\mathbf{k}^{ang}_D {}_C\boldsymbol{\omega}_B + \mathbf{k}^{ang}_P(\mathbf{q}^d_{CB} \boxminus \mathbf{q}_{CB}) \tag{8}$$

for the angular motion. The Jacobian matrices ${}_C\mathbf{J}_P$ and ${}_C\mathbf{J}_R$ are the translational and rotational Jacobian associated to the base expressed in the *Control frame* C, which is a frame aligned with the local estimation of the terrain and with the heading direction of the robot (for more details see [1] and [5]). The $\boxminus$ operator yields the Euler vector which represents the relative orientation between the desired attitude $\mathbf{q}^d_{CB}$ and the estimated one $\mathbf{q}_{CB}$. The gains $\mathbf{k}^{pos}_P$, $\mathbf{k}^{pos}_D$, $\mathbf{k}^{ang}_P$ and $\mathbf{k}^{ang}_D$ are diagonal positive definite matrices of control gains. The reference motion ${}_C\mathbf{r}_{IB}$ and its derivatives are the output of the motion plan which is described in section IV.

Contact force minimization: We minimize the contact forces by setting

$$\begin{bmatrix} \mathbf{0}_{3n_c \times n_u} & \mathbb{I}_{3n_c \times 3n_c} \end{bmatrix} \boldsymbol{\xi}_d = \mathbf{0}. \tag{9}$$

Table I summarizes the tasks and their priority (a lower value indicates a higher priority) used throughout the experiments.

Given a computed set of joint motions and contact forces $\boldsymbol{\xi}_d = \begin{bmatrix} \dot{\mathbf{u}}^T_d & \boldsymbol{\lambda}^T_d \end{bmatrix}^T$, we compute the reference joint torques as $\boldsymbol{\tau}_d = \begin{bmatrix} \mathbf{M}_j & -\mathbf{J}^T_{s_j} \end{bmatrix} \boldsymbol{\xi}_d + \mathbf{h}_j$, where $\mathbf{M}_j$, $\mathbf{J}^T_{s_j}$ and $\mathbf{h}_j$ are defined as in (6).

IV. MOTION OPTIMIZATION

We describe how to obtain a motion plan for the x and y coordinates of the whole-body center of mass (com) such that balance is ensured during locomotion at all times. Fig.3 shows the resulting motion plan obtained during the execution of a dynamic lateral walk.

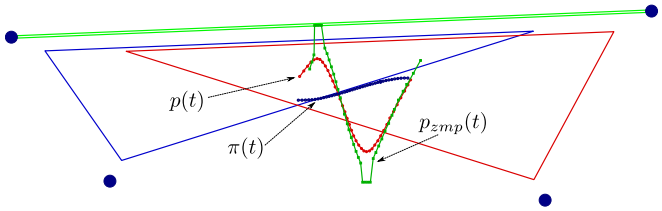

Fig. 3. The solution obtained by the motion planner when a dynamic lateral walk gait is active. The support polygons are computed as the convex hull of the points which are in contact (large blue dots) according to the contact schedule. To increase robustness of the solution, a safety margin is applied to the polygons such that their area is reduced. Only the first three support polygons of the sequence (respectively in red, green and blue) are shown. The resulting center of mass motion $\mathbf{p}(t)$ (red path) is optimized to deviate the least possible from a path regularizer $\boldsymbol{\pi}(t)$ (blue path), while the ZMP $\mathbf{p}_{zmp}(t)$ (green path) is constrained to always lie in the active support polygon. When two lateral legs are simultaneously swinging, the support polygon collapses to a line (shown in green). To avoid numerical issues and excessive accelerations due to a required increase in the order of the approximating spline, we expand the line to a rectangle.

The locomotion framework can receive high-level velocity commands from an external source, i.e. an operator device or a high-level navigation planner. To drive locomotion to the reference speed, footholds are generated for all legs (section IV-A) to obtain the desired average velocity of the torso. This information, together with the footfall pattern (e.g. fig.4), is used to generate a sequence of support polygons (section IV-B) which are sent to the motion plan optimizer (section IV-C). This in turn will produce position, velocity and acceleration profiles for the x and y coordinates of the whole-body center of mass.

The entire plan is computed in the *Plan frame* P which is located at the origin of the inertial frame and follows the yaw rotation of the torso of the robot. Hence, it is always aligned to the heading direction of the robot. This allows us to interpret the problem formulation contributions to the x and y coordinates of the plan as constraints on heading and lateral directions of the motion.

A. Foothold generation

External high-level velocity commands are used to drive locomotion in a specific direction and speed by adjusting the reference footholds, which are computed for each leg at each new control loop. Depending on the contact state of a leg, these are computed in two different ways: when a leg is in contact, the commanded velocities will be projected to foothold locations computed such that the average torso speed matches the required locomotion speed; if a leg is swinging, the reference foothold is computed in the same way but added to a velocity feedback term which stabilizes the robot when it receives a push which produces a velocity control error. For more details, see [5].

B. Support polygon sequence

We describe a gait by defining lift-off ϕ_{lo} and touch-down ϕ_{td} events in the phase domain for each leg. This also defines a contact schedule for all legs. Fig.4 shows the phase events for a dynamic lateral walk, which exhibits overlapping swing

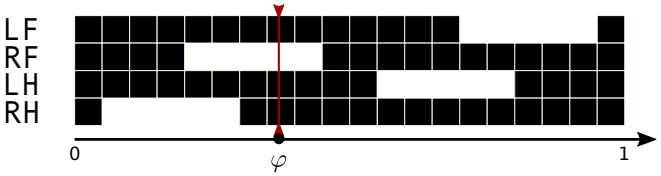

Fig. 4. The contact schedule associated to a dynamic lateral walk. The dark areas represent a contact phase, while the light areas represent a swing phase. The abbreviations stand for left fore (LF), right fore (RF), left hind (LH) and right hind (RH). This gait exhibits an overlap in the swing phases of the left fore/hind and the right fore/hind legs. The red line represents the current phase $\varphi \in [0, 1]$

phases for fore/hind left and fore/hind right legs. Given the touch-down and lift-off events for all legs, as well as the current feet locations and the set of desired footholds, we can compute a sequence of support polygons (defined as the tuple of vertices and time duration in seconds) to be used in the motion planner. We do this whenever a new motion plan is available, such that the new solution adapts to changes in the contact schedule, changes in the reference footholds and changes in the high-level operator velocity. To compute each polygon, we start from the current phase φ and store the $x - y$ coordinates of the feet which are in contact. We search for the next phase event φ_k with $k = 1$ to get the duration of the first polygon $t_0 = t_s(\varphi_k - \varphi)$, with t_s the stride duration in seconds. Thus, we have completely defined a support polygon by its vertices and its duration in seconds. We iterate by updating φ to φ_k and searching for the next phase event. Since the contact schedule is periodic, we repeat these steps until the phase event $\varphi_0 + 1$, which corresponds to the starting phase φ_0.

C. Problem formulation

As done in [19], we choose to represent the motion plan by using two sequences of fifth order polynomial splines for the x and y coordinates of the center of mass position. Each spline has a time duration t_{fk}. Hence, the position $\mathbf{p} \in \mathbb{R}^2$ at time t described by the k-th pair of splines in the sequence can be compactly written as

$$\mathbf{p}(t) = \begin{bmatrix} p_x(t) \\ p_y(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}_{kx}^T \\ \boldsymbol{\alpha}_{ky}^T \end{bmatrix} \boldsymbol{\eta}(t) \tag{10}$$

where

$$\begin{aligned} \boldsymbol{\alpha}_{kx}^T &= \begin{bmatrix} a_{k5}^x & a_{k4}^x & a_{k3}^x & a_{k2}^x & a_{k1}^x & a_{k0}^x \end{bmatrix} \\ \boldsymbol{\alpha}_{ky}^T &= \begin{bmatrix} a_{k5}^y & a_{k4}^y & a_{k3}^y & a_{k2}^y & a_{k1}^y & a_{k0}^y \end{bmatrix} \end{aligned} \tag{11}$$

and

$$\boldsymbol{\eta}(t)^T = \begin{bmatrix} t^5 & t^4 & t^3 & t^2 & t & 1 \end{bmatrix}. \tag{12}$$

Alternatively we can write

$$\mathbf{p}(t) = \begin{bmatrix} \boldsymbol{\eta}(t)^T & \mathbf{0}_{6\times1} \\ \mathbf{0}_{6\times1} & \boldsymbol{\eta}(t)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_{kx} \\ \boldsymbol{\alpha}_{ky} \end{bmatrix} = \mathbf{T}(t)\boldsymbol{\alpha}_k, \tag{13}$$

with $\mathbf{T}(t) \in \mathbb{R}^{2\times12}$. We hence represent the k-th pair of splines as $\mathbf{s}_k^T = \begin{bmatrix} \boldsymbol{\alpha}_{kx}^T & \boldsymbol{\alpha}_{ky}^T \end{bmatrix}$. Using this parametrization, we setup an optimization problem that will solve for all the spline coefficients for $k = 0, \ldots, n-1$, with n the number of splines used to generate the motion. This also allows to easily represent velocities and accelerations as

$$\dot{\mathbf{p}}(t) = \begin{bmatrix} \boldsymbol{\alpha}_{kx}^T \\ \boldsymbol{\alpha}_{ky}^T \end{bmatrix} \dot{\boldsymbol{\eta}}(t) \qquad \ddot{\mathbf{p}}(t) = \begin{bmatrix} \boldsymbol{\alpha}_{kx}^T \\ \boldsymbol{\alpha}_{ky}^T \end{bmatrix} \ddot{\boldsymbol{\eta}}(t) \tag{14}$$

or by writing $\dot{\mathbf{p}}(t) = \dot{\mathbf{T}}(t)\boldsymbol{\alpha}_k$ and $\ddot{\mathbf{p}}(t) = \ddot{\mathbf{T}}(t)\boldsymbol{\alpha}_k$. The problem we solve is then to search for the vector optimal coefficients

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\alpha}_{0x}^T & \boldsymbol{\alpha}_{0y}^T & \boldsymbol{\alpha}_{1x}^T & \boldsymbol{\alpha}_{1y}^T & \ldots & \boldsymbol{\alpha}_{(n-1)x}^T & \boldsymbol{\alpha}_{(n-1)y}^T \end{bmatrix}^T \tag{15}$$

by solving a QP problem. The total number of splines used to approximate the motion is $2n$. The following will describe the setup of an optimization problem formulated as

$$\begin{aligned} \min_{\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{\xi}^T\mathbf{Q}\boldsymbol{\xi} + \mathbf{c}^T\boldsymbol{\xi} \\ \text{s. t.} \quad & \mathbf{A}\boldsymbol{\xi} = \mathbf{b}, \quad \mathbf{D}\boldsymbol{\xi} \leq \mathbf{f}. \end{aligned} \tag{16}$$

In the following, if not explicitly noted otherwise, we will refer only to the x coordinate of each pair of splines $\mathbf{s}_k$. The same arguments hold for the y coordinate.

D. Plan initialization

The motion plan is initialized with an extended state (position, velocity and acceleration) which is used as a hard constraint for the initial state of the spline sequence. This is done by fusing together the previous desired position ${}_C\mathbf{r}_{IB}^{des}$ with the current measured one ${}_C\mathbf{r}_{IB}$ by using an exponentially decaying alpha filter designed as

$${}_C\mathbf{r}_{IB}^{ref} = \alpha {}_C\mathbf{r}_{IB}^{des} + (1 - \alpha) {}_C\mathbf{r}_{IB} \tag{17}$$

with $\alpha = 0.5\mathrm{e}^{-\lambda t_c}$, where $\boldsymbol{\lambda} \in \mathbb{R}$ can be set to weigh the desired position as a function of the computation time t_c since the last successful optimization. A similar filter is employed to set the initial velocities, whereas acceleration is initialized with the last available reference one.

E. Cost function

In our formulation, several terms contribute to the Hessian matrix $\mathbf{Q} \in \mathbb{R}^{12n\times12n}$ and the linear term $\mathbf{c} \in \mathbb{R}^{12n}$ appearing in (16). We penalize the deviation of the position $\mathbf{p}(t) = \mathbf{T}(t)\boldsymbol{\alpha}$ from a desired position reference $\mathbf{p}_r$ by writing

$$\begin{aligned} &\frac{1}{2}\|\mathbf{T}\boldsymbol{\alpha} - \mathbf{p}_r\|_{\mathbf{W}}^2 \\ &= \frac{1}{2}(\mathbf{T}\boldsymbol{\alpha} - \mathbf{p}_r)^T\mathbf{W}(\mathbf{T}\boldsymbol{\alpha} - \mathbf{p}_r) \\ &= \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{T}^T\mathbf{W}\mathbf{T}\boldsymbol{\alpha} - \mathbf{p}_r^T\mathbf{W}\mathbf{T}\boldsymbol{\alpha} + \frac{1}{2}\mathbf{p}_r^T\mathbf{W}\mathbf{p}_r. \end{aligned} \tag{18}$$

Minimizing the squared norm in (18) is equivalent to solving a QP in the form 16 with

$$\mathbf{Q} = \mathbf{T}^T\mathbf{W}\mathbf{T} \qquad \mathbf{c} = -\mathbf{T}^T\mathbf{W}^T\mathbf{p}_r. \tag{19}$$

To penalize deviation for velocity references, it is simply required to use $\dot{\mathbf{T}}$ and $\dot{\mathbf{p}}_r$ in (18). A similar reasoning can be done for acceleration deviations.

1) Acceleration minimization: As shown in [10], we can minimize the acceleration by writing

$$\mathbf{Q}_k^{acc} = \begin{bmatrix} (400/7)t_f^7 & 40t_f^6 & 24t_f^5 & 10t_f^4 & \\ 40t_f^6 & 28.8t_f^5 & 18t_f^4 & 8t_f^3 & \vdots \\ 24t_f^5 & 18t_f^4 & 12t_f^3 & 6t_f^2 & \mathbf{0}_{4\times 2} \\ 10t_f^4 & 8t_f^3 & 6t_f^2 & 4t_f & \vdots \\ \cdots & \mathbf{0}_{2\times 4} & \cdots & & \mathbf{0}_{2\times 2} \end{bmatrix} \quad (20)$$

with a null linear term $\mathbf{c}_k^{acc} = \mathbf{0}$. Note that if this were the only term added to the cost function, there would be no cost associated to the α_{1k} and α_{2k} coefficients of each spline, leading to a positive semi-definite Hessian. This is problematic when using a method such as the Active Set one [6] which requires the Hessian to be positive definite. In that case, a regularizer term can be added as

$$\mathbf{Q}_k^{acc_\rho} = \begin{bmatrix} \mathbf{0}_{4\times 4} & \mathbf{0}_{4\times 2} \\ \mathbf{0}_{2\times 4} & \rho\mathbb{I}_{2\times 2} \end{bmatrix} \quad (21)$$

with $\rho = 10\mathrm{e}^{-8}$ and a null linear term.

2) Soft final constraints: We set the final position $\mathbf{p}_f \in \mathbb{R}^2$ to be at the center $\mathbf{p}_f^{ref} \in \mathbb{R}^2$ of the polygon defined by the planned footholds, which is the polygon that would support the robot if it would stop to stand at the end of the support polygon sequence. To minimize the norm $\|\mathbf{p}_f - \mathbf{p}_f^{ref}\|_{\mathbf{W}_f}^2 = \|\mathbf{A}_f\mathbf{s}_f - \mathbf{p}_f^{ref}\|_{\mathbf{W}_f}^2$ we write

$$\mathbf{Q}_f = \mathbf{A}_f^T\mathbf{W}_f\mathbf{A} \qquad \mathbf{c}_f = -\mathbf{A}^T\mathbf{W}_f\mathbf{p}_f^{ref}, \quad (22)$$

with $\mathbf{W}_f \in \mathbb{R}^{2\times 2}$ a symmetric diagonal matrix of positive weights. To avoid solutions in which the optimizer places the final state far away from the reference position, we add inequality constraints on the position such that it is constrained in a box centered at the reference position.

3) Deviation from previous solution: Since we are computing a new optimization as soon as the previous one was successful, to avoid large deviations between successive motion plans we penalize deviations on position, velocity and acceleration obtained by the current solution $\boldsymbol{\xi}$ from the ones resulting from the previous one $\boldsymbol{\xi}_{i-1}$. Given t_{pre} the time elapsed since the last successful optimization, we penalize the deviations $\|\mathbf{p}(\bar{t}) - \mathbf{p}_{i-1}(t_{pre} + \bar{t})\|_{\mathbf{W}_{pre}}^2, \quad \forall \bar{t} \in [0, t_f]$, where $t_f = \sum_{k=0}^{n-1} t_{fk}$ is the sum of the durations of all of the n splines that compose the motion. We discretize the optimization horizon $[0, t_f]$ using a sample time dt. We employ a similar cost penalizing deviations from velocities and accelerations obtained by the last solution.

4) Path regularization: Continuous update of the motion plan can cause drift in the motion of the torso w.r.t. the reference footholds. This can be caused by the accumulation of control errors, which alter the solution such that the motion can become unfeasible. To avoid this issue, we add a cost on the deviation from a reference *regularizer path*. The path itself is approximated as a sequence of splines represented by $\boldsymbol{\pi}(t)$, $\dot{\boldsymbol{\pi}}(t)$ and $\ddot{\boldsymbol{\pi}}(t)$. The spline coefficients of the path regularizer are obtained from a minimization problem setup such that:

- The initial state $\boldsymbol{\pi}(0)$, $\dot{\boldsymbol{\pi}}(0)$ and $\ddot{\boldsymbol{\pi}}(0)$ coincides with the center of the initial support polygon and is equal to the initial velocity and acceleration
- The final state $\boldsymbol{\pi}(t_f)$, $\dot{\boldsymbol{\pi}}(t_f)$ and $\ddot{\boldsymbol{\pi}}(t_f)$ is set as in section IV-E.2
- Acceleration is minimized
- Splines are smoothly connected by setting equality constraints on the state at the spline junctions

By sampling the entire motion as done in section IV-E.3, we penalize deviations of the resulting motion from the path regularizer.

F. Equality constraints

Since the entire motion is composed of a sequence of splines, we setup the problem to ensure that they are connected. Since the initial state cannot be modified by the motion plan, we set a hard equality constraint such that the initial state of the first spline $\mathbf{s}_0$ coincides with the one set in the plan initialization. The initial hard constraint can be written as $\mathbf{p}(0) = \mathbf{p}^r$, $\dot{\mathbf{p}}(0) = \dot{\mathbf{p}}^r$ and $\ddot{\mathbf{p}}(0) = \ddot{\mathbf{p}}^r$. Hence, we write

$$\begin{bmatrix} \boldsymbol{\eta}(0)^T \\ \dot{\boldsymbol{\eta}}(0)^T \\ \ddot{\boldsymbol{\eta}}(0)^T \end{bmatrix} \boldsymbol{\alpha}_0^x = \begin{bmatrix} \mathbf{p}_x^r \\ \dot{\mathbf{p}}_x^r \\ \ddot{\mathbf{p}}_x^r \end{bmatrix}. \quad (23)$$

To ensure that two splines $\mathbf{s}_k$ and $\mathbf{s}_{k+1}$ are connected, we constrain them such that

$$\begin{bmatrix} \boldsymbol{\eta}(t_{fk})^T & -\boldsymbol{\eta}(0)^T \\ \dot{\boldsymbol{\eta}}(t_{fk})^T & -\dot{\boldsymbol{\eta}}(0)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_k^x \\ \boldsymbol{\alpha}_{k+1}^x \end{bmatrix} = \mathbf{0}, \quad (24)$$

with t_{fk} representing the duration in seconds of spline $\mathbf{s}_k$. When the equality junction connecting two splines lies between two disjoint support polygons, we cannot guarantee the smoothness or the motion anymore, but will have to allow the ZMP to jump, which will cause a jump in the acceleration reference. Although this has a negative effect on the controller, it does allow the optimizer to find a solution when traversing disjoint support polygons. We check whether two polygons intersect by using the *Separating Axis Theorem* (SAT) described in [7]. If such an intersection exists, we also constrain acceleration between two splines by writing

$$\begin{bmatrix} \ddot{\boldsymbol{\eta}}(t_f)^T & -\ddot{\boldsymbol{\eta}}(0)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_k^x \\ \boldsymbol{\alpha}_{k+1}^x \end{bmatrix} = \mathbf{0}. \quad (25)$$

G. Inequality constraints

As demonstrated in [11] and [19], balance during locomotion can be ensured by constraining the *Zero-Moment Point* (ZMP) to lie inside the support polygon. Starting from the measured feet configuration at the time the motion plan is called and using the planned feet positions, we compute a sequence of support polygons and their duration by using the contact schedule as defined in [5]. By assuming at least two feet on the ground at all times, the support polygon for a quadrupedal robot can be a line, a triangle or a quadrilateral.

As discussed in [16] and [19], the location of the ZMP is a function of the motion of the center of mass. The location of the x coordinate of the ZMP is defined by

$$x_{zmp} = x_{com} - \frac{z_{com}\ddot{x}_{com}}{g + \ddot{z}_{com}}, \quad (26)$$

where g is the gravitational acceleration. We reorder the vertices of each support polygon counter-clockwise by computing their polar coordinates w.r.t. the center of the support polygon and comparing their phase. We can then constrain the ZMP by adding one constraint for each edge of the support polygon by writing $ax_{zmp} + by_{zmp} + c \geq 0$, where a, b and c are the coefficients of the line that passes through the vertices of an edge of the support polygon. The normal vector to this line $\mathbf{n} = \begin{bmatrix} a & b \end{bmatrix}^T$ is directed towards the inside of the polygon.

As mentioned in section IV-E, we place additional inequality constraints on the final position $\mathbf{p}_f$ of the motion. This is obtained by adding four constraints of the form (26) on $\mathbf{p}_f$.

H. Constraint relaxation

The optimizer can fail if the constraints are too tight. First, the initial ZMP may not lie in the current support polygon. For this reason, we exclude the ZMP constraint on the first sample of the motion plan. This way the optimizer is still able to find a solution, and our experiments have shown this to work on the real system.

Second, the support polygon safety margin may be too high. To counteract this, whenever an optimization fails, we iteratively decrease the margin by a fixed amount, and we increase it back at a slower rate to ensure the success of the optimization.

Finally, we check for the time duration t_{fk} associated to each polygon. If it is smaller or comparable to the sample time used to discretize the motion and to setup the constraints, we remove the support polygon from the sequence.

V. Experiments

A. Setup

Our experiments[1] were conducted on ANYmal [9], an accurately torque-controllable quadrupedal robot. Control signals are generated in a 400Hz control loop which runs on a dedicated on-board computer together with state estimation [2] and communication with the drives. For modeling and computation of kinematics and dynamics, we use the open-source Rigid Body Dynamics Library [13] (RBDL), which is a C++ implementation of the algorithms described in [4]. To numerically solve the QP problems described in the previous sections, we use a custom version of the QuadProg++ [12] library, a C++ open-source QP solver which implements the Active Set algorithm described in [6].

B. Dynamic locomotion

We have tested our planning and control framework by executing several gaits, including a dynamic lateral walk with overlapping swing phases; a trotting gait; a pacing gait; a transition between dynamic lateral walk and pacing (see Fig.7). The execution of all these gaits was successful both on flat terrain and on slightly uneven terrain, including locomotion over a small but unperceived step. We have also tested an online transition from stand to a dynamic lateral walk to a pace. This was achieved by applying a transition from the walking to the pacing gait by interpolating the two footfall patterns as a function of the commanded heading speed. As shown in the videos, we can smoothly switch from stand, to dynamic walk to pace and safely back to stand.

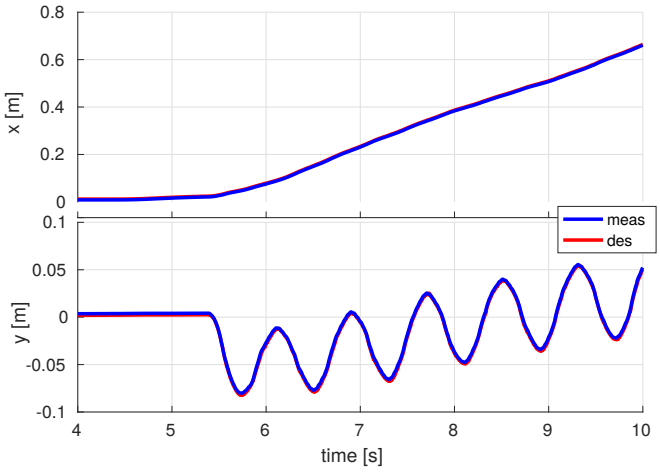

Fig. 5. The time evolution of the measured (blue) and reference (red) whole-body center of mass position on the x and y directions during a pacing gait.

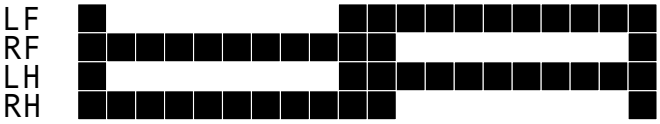

Fig. 6. The contact schedule associated to a pacing gait. This gait exhibits a complete overlap in the swing phase of two legs which are on the same side of the main body, namely fore/hind left or fore/hind right legs.

VI. Conclusions and Future Work

We have shown how it is possible to execute highly dynamic and agile gaits on a torque-controllable quadruped robot by combining a motion planner based on a simplified dynamic model and a hierarchical whole-body controller. The motion planner based on the ZMP stability criterion is continuously updating the reference trajectory as a function of the current gait pattern and the robot state; the whole-body controller tracks the reference plan by solving a hierarchy of tasks. We have demonstrated the execution of dynamic gaits such as a trot, a dynamic lateral walk, a pace and a transition between the last two on ANYmal. To the best of our knowledge, this is the first time that experimental results are published for such gaits.

The next steps will be focused on improving the main features of gait execution and gait transition. This includes foothold evaluation and gait pattern adaptation in order to be able to deal with harsh changes in the state of the robot and sudden changes in the active gait pattern. The framework should also be extended to include full flight phases, which would allow to perform even more agile locomotion.

[1] https://youtu.be/iUQE-ZQqdJY

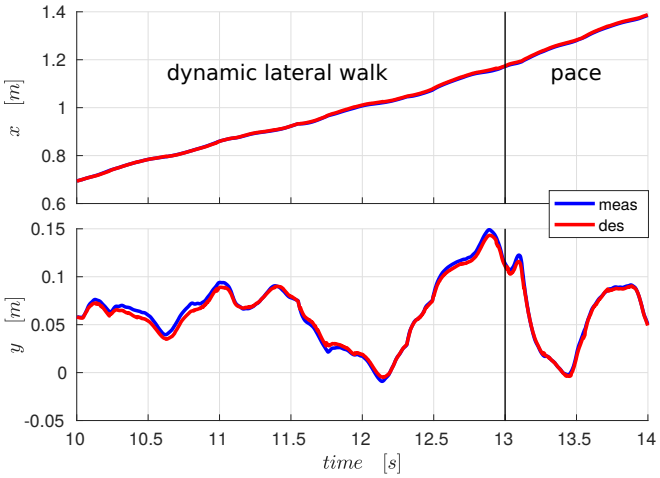

Fig. 7. The time evolution of the measured and reference whole-body center of mass position in the inertial frame during a transition from dynamic lateral walk to pace.

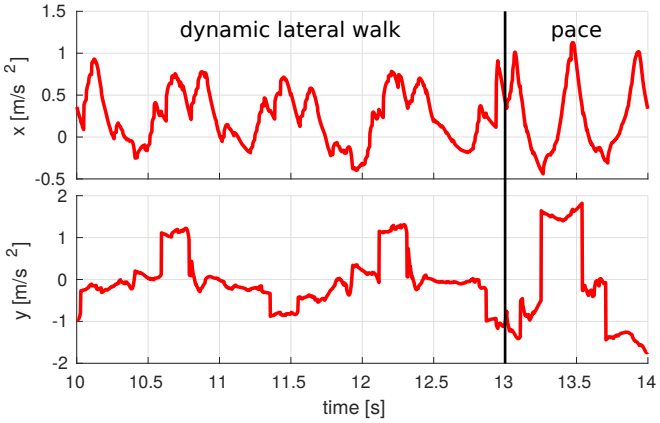

Fig. 8. The time evolution of the reference acceleration for the whole-body center of mass during a transition from dynamic lateral walk to pace.

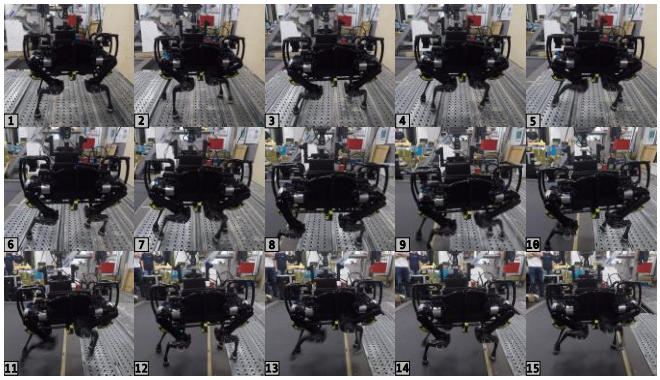

Fig. 9. A sequence showing ANYmal during a transition from a dynamic lateral walk to a pacing gait. Balance is retained even though there is an unperceived step.

References

[1] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter. Perception-less terrain adaptation through whole body control and hierarchical optimization. In *2016 IEEE-RAS 16th Int. Conf. on Humanoid Robots (Humanoids)*, pages 558–564, Nov 2016.

[2] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart. State estimation for legged robots on unstable and slippery terrain. In *2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 6058–6064. IEEE, nov 2013.

[3] P. Fankhauser, C. D. Bellicoso, C. Gehring, R. Dub, A. Gawel, and M. Hutter. Free gait - an architecture for the versatile control of legged robots. In *2016 IEEE-RAS 16th Int. Conf. on Humanoid Robots (Humanoids)*, pages 1052–1058, Nov 2016.

[4] R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer US, Boston, MA, 2008.

[5] C. Gehring, S. Coros, M. Hutler, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, and R. Siegwart. Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot. *IEEE Robotics Automation Magazine*, 23(1):34–43, March 2016.

[6] D. Goldfarb and a. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27(1):1–33, 1983.

[7] S. Gottschalk. Separating axis theorem. Technical report, TR96-024, Department of Computer Science, UNC Chapel Hill, 1996.

[8] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal. Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 981–988, Sept 2014.

[9] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, P. Fankhauser, M. Bloesch, R. Diethelm, and S. Bachmann. ANYmal - A Highly Mobile and Dynamic Quadrupedal Robot. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[10] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal. Fast, robust quadruped locomotion over challenging terrain. In *2010 IEEE Int. Conf. on Robotics and Automation*, pages 2665–2670. IEEE, may 2010.

[11] 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*, 30(2):236–258, nov 2010.

[12] Luca Di Gaspero. QuadProg++. Available at `http://quadprog.sourceforge.net/`, 1998.

[13] Martin Felis. Rigid Body Dynamics Library. Available at `http://rbdl.bitbucket.org/`.

[14] R. McGhee and A. Frank. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3:331–351, aug 1968.

[15] H.-W. Park, P. M. Wensing, and S. Kim. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. *Robotics: Science and Systems*, 2015.

[16] P. Sardain and G. Bessonnet. Forces acting on a biped robot. center of pressure-zero moment point. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 34(5):630–637, Sept 2004.

[17] 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. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(6):831–849, 2011.

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

[19] A. W. Winkler, C. Mastalli, M. Focchi, D. G. Caldwell, and I. Havoutis. Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain. *Icra*, pages 5148–5154, 2015.