Line Walking and Balancing for Legged Robots with Point Feet

Carlos Gonzalez, Victor Barasuol, Marco Frigerio, Roy Featherstone,
Darwin G. Caldwell, and Claudio Semini

To be published in:

Supplementary Material:

Video of experimental and simulation results: https://youtu.be/gbon2Bde25g

To cite this paper:

C. Gonzalez, V. Barasuol, M. Frigerio, R. Featherstone, D.G. Caldwell and C. Semini, "Line Walking and Balancing for Legged Robots with Point Feet," *In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, accepted.

For this and other publications from the Dynamic Legged Systems lab (DLS):
https://dls.iit.it/dls-publications

Line Walking and Balancing for Legged Robots with Point Feet

Carlos Gonzalez[1], Victor Barasuol[1], Marco Frigerio[2], Roy Featherstone[3],
Darwin G. Caldwell[3], Claudio Semini[1]

Abstract— **The ability of legged systems to traverse highly-constrained environments depends by and large on the performance of their motion and balance controllers. This paper presents a controller that excels in a scenario that most state-of-the-art balance controllers have not yet addressed: line walking, or walking on nearly null support regions. Our approach uses a low-dimensional virtual model (2-DoF) to generate balancing actions through a previously derived four-term balance controller and transforms them to the robot through a derived kinematic mapping. The capabilities of this controller are tested in simulation, where we show the 90kg quadruped robot HyQ crossing a bridge of only 6 cm width (compared to its 4 cm diameter spherical foot), by balancing on two feet at any time while moving along a line. Additional simulations are carried to test the performance of the controller and the effect of external disturbances. Lastly, we present our preliminary experimental results showing HyQ balancing on two legs while being disturbed.**

I. Introduction

Legged robotic systems are becoming increasingly versatile and their capabilities are being evaluated in fields such as inspection and disaster relief, among others. The usability of these systems, however, is compromised when the robot encounters an environment with limited support contacts for the feet. In these scenarios, the balancing capabilities of the robot are crucial for the completion of the task.

Consider the scenario of crossing a narrow bridge, as shown in Fig. 1. In this case, a high-fidelity balance controller is required. Most of today's balance controllers, however, would fail this task since line walking on a nearly null support polygon is required. In this work, we consider the possibility of achieving this task (crossing a narrow bridge) using a momentum-based balance controller applied on a low-dimensional virtual model that controls the physical process of balancing.

This work is based on the balance controller of [1], which was originally for planar systems, but in this paper we will extend this approach to the control of a 12 degrees of freedom (DoF) quadruped, which will balance on two point feet. In addition, we will further extend the ideas previously published in [1] to allow motions along a contact line in 3-dimensions. To the best of our knowledge, this is the first time that such a task has been achieved in simulation and that a heavy legged robot has balanced on a null support polygon experimentally without continuously taking steps.

The main contributions of this work are (a) the derivations of the kinematic mapping relating the planar model and a quadruped, (b) the usage of the extended balance controller in conjunction with an external motion generator to create balanced walking motions, (c) simulation experiments showing the performance of the controller tracking an angular position reference on a quadruped, (d) simulations showing for the first time a quadruped robot walking along a line on two point feet, and (e) experimental validation of the proposed balance controller showing the hydraulically actuated quadruped robot HyQ [2] balancing on a support line.

The presentation of this work is as follows: related work is presented in Sec. II; Section III summarizes the main concepts of the four-term balance controller which is described in detail in [1]. Section IV presents the step-by-step procedure to obtain the kinematic mapping that extends the controller to the quadruped case. Section V presents the design of the combined motion and balance controller to achieve locomotion. The simulations and experimental results are presented in Sec. VI, and the concluding remarks and topics of future work are presented in Sec. VII.

II. Related Work

Tasks such as balancing on a line or on a single point have been previously studied on low dimensional systems. Some examples include Cubli, the cube that balances and stands on a single point [3], self-balancing bicycles [4], mobile robots balancing on a ball [5], and Tippy, a high-performance balancing robot controlled by only one actuator [6]. The balance controllers in the first two systems are similar in that their objective is to bring the systems to a completely vertical position by controlling the rotor velocity of fly wheels. On the other hand, the balance controller of the robot balancing on a ball achieves this by tracking the desired projection of

[1]Dynamic Legged Systems Lab, Istituto Italiano di Tecnologia, Genoa, Italy. Email: <first name>.<last name>@iit.it
[2]Department of Mechanical Engineering, KU Leuven, Belgium. Email: marco.frigerio@kuleuven.be
[3]Department of Advanced Robotics, Istituto Italiano di Tecnologia, Genoa, Italy. Email: <first name>.<last name>@iit.it
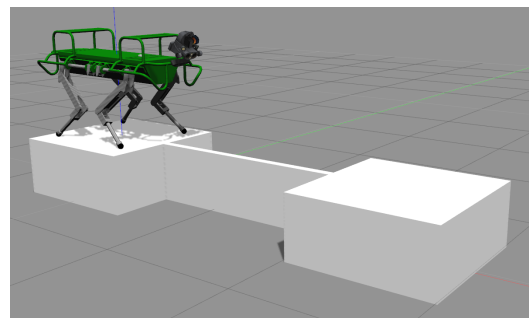
Fig. 1. Sample scenario where a high-fidelity balance controller is required. The length and width of the bridge are 1.5 m and 6 cm, respectively.

the center of mass (CoM) of the robot using a PID controller. The balance controller used on Tippy is based on [1] and it differs from the previously mentioned controllers in that its goal is to track a reference angular trajectory while balancing and in that the states of the controlled system relate to a plant describing the physical process of balancing, rather than directly controlling position and velocity states.

Generalizing the balance controllers of these systems to higher dimensional systems is not a trivial task. Some of the current approaches to achieve balancing seek to control the full dimensional robot by controlling a low dimensional virtual model while considering the robot's centroidal dynamics [7]–[9]. In [7] a double inverted pendulum model is used to represent the actuated legs and torso of a bipedal robot, for which combined efforts to control the ankle and posture are synthesized. This effectively emulates a combined ankle and hip strategy for balancing. Other methods to achieve a balanced state consist of taking planned steps in locations determined by a simplified gait model, often also taking the form of an inverted pendulum [8], [9]. These approaches either make use of the point mass assumption, hence limiting control over angular momentum, or assume the system has access to a base of support and, thus, can generate ground reaction moments through the stance feet.

Other researchers have approached the problem of balancing by considering the full joint space dynamics. These approaches often use the principles of the nominal work of [10] on momentum-based balance control, which solves for desired joint motions to track specified linear and angular momenta. Several extensions and modifications have also been considered. The approach in [11] instead proposes a force control policy that considers the coupling between the linear and angular momentum and finds appropriate torques satisfying a hierarchy of tasks by solving a sequence of prioritized quadratic programs (QPs). Furthermore, the authors validate their approach experimentally on a torque-controlled biped. In [12] the authors opted for structuring the hierarchies of the tasks through null space projections rather than solving a series of QPs and solved an optimization problem only when determining the distribution of the balancing wrenches.

Another QP-based approach that showed a similarly challenging balancing scenario based on partial footholds was presented in [13]. In this work, the authors designed a method to detect partial footholds, i.e., foot locations in which only part of the foot is in contact with the ground, and incorporated this information into their optimization problem to design trajectories that successfully bring the robot to the next foothold without falling. This allows the robot to stand on a line contact momentarily while the swing leg is reaching for the next foothold to regain a double stance where a non-empty support polygon allows the robot to recover its balance. In contrast, in this work we present a controller that not only allows the robot to balance, but also to linger, and move along a contact line, presenting a more challenging task for the balance controller.

The aforementioned approaches work well on systems with a finite support polygon, but are impractical when balancing on a line as they rely on controlling the center of pressure (CoP) within the polygon, which is now reduced to a line. Similarly, controllers largely relying on an inaccurate CoM can drive the robot to an unbalanced configuration.

For other multi-legged robots, the *crawl* often constitutes the most balanced locomotion strategy. It is typically separated into two stages. One stage defines a desired gait sequence and the other plans a path for the projection of the CoM (or, alternatively, for the zero-moment point, if this is used as the dynamic stability criterion) such that it traverses the sequence of support triangles formed by the desired gait sequence [14], [15]. These approaches evidently make use of an available support polygon. A more dynamic motion which is not based on support polygons is the *trot*, which involves stance phases with only two feet in contact with the ground (thus a series of line contacts). However, these rely on the next stance happening soon and can still cause the robot to fall if the gait frequency is too slow.

More recently, an approach to balance a quadruped robot on two feet was proposed in [16]. The approach therein uses linearization techniques on the centroidal dynamics of the robot to pose an optimal control problem subject to friction constraints, whose solution is approximated by solving a QP. Our proposed approach is inherently different, mainly in that we control the process of balancing and the impedance associated to the virtual model being used. This allows us to exploit traditional techniques to perform additional tasks, such as walking along a line while balancing.

III. THE BALANCE CONTROLLER

This section briefly summarizes the key concepts of the balance controller presented in [1]. Fig. 2 illustrates the plant describing the balancing behavior of a 2-DoF planar inverted pendulum, which swings about a pivot point on the ground (first degree of freedom) by controlling the revolute joint connecting its two links (second degree of freedom). The states of the plant are the angular momentum of the pendulum about its pivoting point, L, its first and second time-derivatives, $\dot{L}$ and $\ddot{L}$, and the position of the actuated joint used for balancing, q_2. In order to achieve a balanced state, the controller must regulate the angular momentum and its derivatives, and lead the joint used for balancing to its desired value.

For any planar system balancing on a point, there are two values that reveal its balancing behavior: the velocity gain G_v and the time constant of toppling T_c [17]. The former quantifies the extent to which motion in the actuated joint
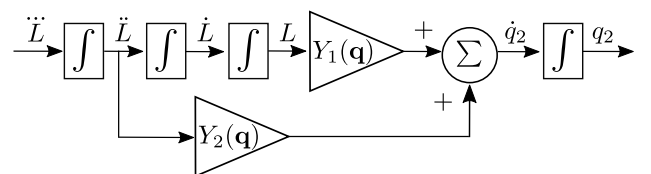

Fig. 2. Plant describing the exact physics of the process of balancing for a 2-DoF system

affects the horizontal motion of the CoM. Hence, a small G_v indicates a limited ability to control and move the CoM on top of the pivoting point, which is required to balance. The T_c value corresponds to the time constant of the robot falling if none of its joints moved. These two values relate to the gains in the model of the physical process of balancing. The particular relationship between these values is given by $Y_1(\mathbf{q}) = \frac{1}{mgT_c^2 G_v}$ and $Y_2(\mathbf{q}) = -\frac{1}{mgG_v}$, where m and g are the total mass of the system and the magnitude of the acceleration due to gravity, respectively, and $\mathbf{q} \in \mathbb{R}^n$ is the vector of joint positions of the system with n degrees of freedom. Note that these definitions hold for a generic n-dimensional planar system.

After defining the parameters of the plant describing the process of balancing, a proper controller can be designed. In [1] such a controller is designed by using pole placement on the plant assuming the parameters Y_1 and Y_2 are constant.

It can be shown that the linear plant has a zero in the right-hand side of the complex plane, which implies that it exhibits non-minimum-phase behavior. Such behavior is intrinsic to the physics of balancing, and cannot be overcome by any control system. However, it is possible to compensate for this behavior by filtering the commanded signal with a low-pass filter running backwards in time (i.e., from the future to the present). This effectively eliminates the non-minimum phase behavior from the transfer function of the closed-loop system, resulting in motion in which the robot *leans in anticipation* of future commanded motions. The interested reader is referred to [1] for more details.

IV. KINEMATIC MAPPING

To exploit the balance controller on a quadruped robot, we define the motion constraints that define a lower dimensional operational space, which corresponds to the two degrees of freedom of the pendulum. We call *kinematic mapping* the function transforming the state variables from the quadruped to the pendulum space. The mapping allows us to apply the controller in pendulum space, and convert its output into signals suitable for the actual robot. The derivation of the mapping is described in the following paragraphs.

First, consider a quadruped standing on a single pair of diagonal legs, with the other leg pair retracted (lifted up from the ground), as shown in Fig. 3(a). Project a *virtual* inverted pendulum moving in the vertical plane perpendicular to the line of contact defined by the two stance feet. The pendulum consists of two links, the leg and the torso, and two revolute joints, the pivot and the hip (see Fig. 3(b)). The pivot joint is passive and its axis coincides with the line of contact of the quadruped; the hip joint is actuated and its axis is parallel to the previous one, at a distance equal to the length of the virtual leg link.

The virtual torso link is composed of the real robot torso and the retracted legs, thus they have the same inertia and undergo the same motion (see Fig. 3(a)). We attach three reference frames to the virtual pendulum: vbase is an inertial frame with origin at the pivot contact point with its x-axis coinciding with the pivot axis; vleg is a frame attached

to the virtual leg and also its x-axis coincides with the pivot axis; vtorso is attached to the virtual torso and its x-axis coincides with the virtual hip axis. The origins of vtorso and of the real torso frame (rtorso) coincide, and their z-axes are aligned. The frames only differ by a relative rotation of ϕ_t radians about their local z-axis.

The states of the virtual model are $\mathbf{y} = [y_p \ y_h]^T$ and $\dot{\mathbf{y}} = [\dot{y}_p \ \dot{y}_h]^T$, where p and h stand for pivot and hip, respectively. The relation between the virtual states $(\mathbf{y}, \dot{\mathbf{y}})$ and the robot states $(\mathbf{q}, \dot{\mathbf{q}})$ is obtained in two steps. First, the sensor data obtained from the robot is used to compute the corresponding states of the virtual model (Sec. IV-A). Then, the kinematic mapping is derived through a concatenation of derived Jacobians (Sec. IV-B).

A. Sensor data to virtual model states

Sensors on the robot measure the joint states $\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^{12}$, the angular velocity of the robot torso, $\boldsymbol{\omega} \in \mathbb{R}^3$, and a unit vector pointing up, $\overrightarrow{\mathbf{up}}$, both in rtorso coordinates. The stance feet position, $\mathbf{p}_{f_i} \in \mathbb{R}^3$ $i \in \{1, 2\}$, provide the angle of the support line:

$$\phi_t = \tan^{-1}\left(\frac{p_{f_1,y} - p_{f_2,y}}{p_{f_1,x} - p_{f_2,x}}\right) \quad (1)$$

where the second subscript of the foot position indicates the specific coordinate, e.g., $p_{f_1,y}$ is the y-coordinate of the first stance foot. From ϕ_t we can compute the rotation matrix that transforms 3D vectors from rtorso to vtorso coordinates, $\mathbf{E}_{\phi_t} \in SO(3)$. At this point, simple geometric observations yield some of the relations between sensor data and the state of the virtual pendulum. For example, we have that

$$\dot{y}_p + \dot{y}_h = (\mathbf{E}_{\phi_t}\boldsymbol{\omega})_x \quad (2)$$

where $(\cdot)_x$ indicates the x-component of the vector inside the parenthesis; we also have that

$$y_p + y_h = \sin^{-1}\left(\mathbf{E}_{\phi_t}\overrightarrow{\mathbf{up}}\right)_y \quad (3)$$
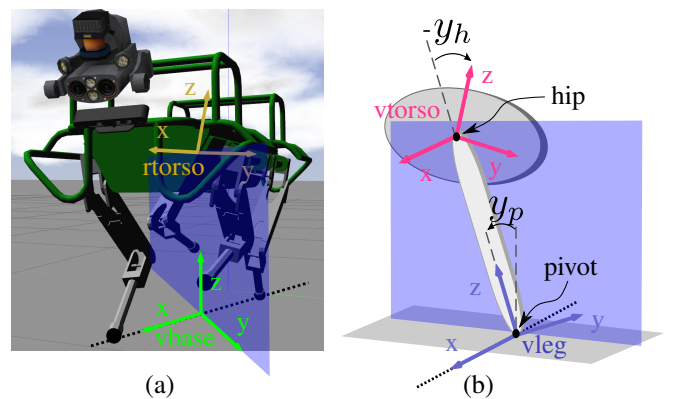

(a)   (b)

Fig. 3. Inertial and link frames of the virtual model in relation to the real robot. (a) The location of the stance feet dictate the axis about which the joints of the virtual model rotate (dotted line), which is orthogonal to the plane where the virtual model lies. (b) Location of coordinate frames of the virtual model and its corresponding joint variables y_p and y_h.

With reference to Fig. 3(b), we also observe that the pivot point in the `vtorso` frame, denoted by (p_y, p_z), allows us to calculate the virtual hip joint angle:

$$y_h = \tan^{-1}\left(\frac{p_y}{p_z}\right) \tag{4}$$

and by differentiating (4) w.r.t. time, it follows that

$$\dot{y}_h = \frac{p_z v_y - p_y v_z}{p_y^2 + p_z^2} \tag{5}$$

Hence, at each time instant, the virtual states $\mathbf{y}$ and $\dot{\mathbf{y}}$ are found by solving (2) - (5).

B. Explicit motion constraints

The purpose of this phase is to determine the coefficients of the following motion constraint equations:

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{G}_j \dot{\mathbf{y}} \\ \ddot{\mathbf{q}} &= \mathbf{G}_j \ddot{\mathbf{y}} + \mathbf{g}_j \end{aligned} \tag{6}$$

where $\mathbf{G}_j \in \mathbb{R}^{n\times 2}$ and $\mathbf{g}_j = \dot{\mathbf{G}}_j \dot{\mathbf{y}}$ relate the motion of the pendulum to the exact desired motion of the real robot (cf. Sec. 3.2 in [18]). $\mathbf{G}_j$ and $\mathbf{g}_j$ also allow us to map the dynamics quantities of the robot, like the inertia matrix, into the corresponding quantity of the virtual pendulum, in order to apply the balance controller. Thus, the $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ in (6) represent *desired* motions the robot should track in order to actually mimic the motion of the inverted pendulum. We will therefore use these values in our control law.

The derivation of $\mathbf{G}_j$ and $\mathbf{g}_j$ is done in stages, as described next.

1) Virtual pendulum to robot torso: In the first stage, we look into the relation between the virtual joint velocity $\dot{\mathbf{y}}$ and the real torso velocity $\mathbf{v}_t$, and we find the matrices $\mathbf{G}_t \in \mathbb{R}^{6\times 2}$ and $\mathbf{g}_t \in \mathbb{R}^6$ such that

$$\mathbf{v}_t = \mathbf{G}_t \dot{\mathbf{y}} \tag{7a}$$
$$\dot{\mathbf{v}}_t = \mathbf{G}_t \ddot{\mathbf{y}} + \mathbf{g}_t \tag{7b}$$

Simple observation on the kinematic model of the pendulum reveals that $\mathbf{v}_t$ can be written as follows:

$$\mathbf{v}_t = \mathbf{S}_p \dot{y}_p + \mathbf{S}_h \dot{y}_h \tag{8}$$

where $\mathbf{S}_p$ and $\mathbf{S}_h$ are the motion subspace matrices for the pivot and hip joints, respectively. It follows that

$$\mathbf{G}_t = \begin{bmatrix} \mathbf{S}_p & \mathbf{S}_h \end{bmatrix} \tag{9}$$

To obtain $\mathbf{g}_t$, we have to compute the time derivative of $\mathbf{G}_t$ relative to a *stationary* coordinate frame. Since the pivot axis is fixed relative to a stationary frame, $\dot{\mathbf{S}}_p = \mathbf{0}$. On the other hand, the hip joint axis does move in space due to the pivot motion, thus $\mathbf{S}_h$ is changing according to

$$\dot{\mathbf{S}}_h = \mathbf{S}_p \dot{y}_p \times \mathbf{S}_h \tag{10}$$

(cf. [18]) and thus,

$$\mathbf{g}_t = (\mathbf{S}_p \dot{y}_p \times \mathbf{S}_h)\dot{y}_h \tag{11}$$

Note that, in the equations above, we omitted the coordinate transformation matrices, as they do not change the algorithm but make the notation less clear.

2) Torso to feet: The goal is now to map the velocity of the torso to the velocity of the stance feet, relative to the origin of the torso frame itself. In this phase we assume that the stance feet are fixed on the ground. For each stance foot $i \in \{1, 2\}$, we seek the transformation matrices $\mathbf{G}_{f_i} \in \mathbb{R}^{3\times 6}$ and $\mathbf{g}_{f_i} \in \mathbb{R}^3$ such that

$$\mathbf{v}_{f_i} = \mathbf{G}_{f_i} \mathbf{v}_t \tag{12a}$$
$$\mathbf{a}_{f_i} = \mathbf{G}_{f_i} \dot{\mathbf{v}}_t + \mathbf{g}_{f_i} \tag{12b}$$

where $\mathbf{g}_{f_i} = \dot{\mathbf{G}}_{f_i} \mathbf{v}_t$. Note that $\mathbf{v}_{f_i}$ and $\mathbf{a}_{f_i}$ are 3D Euclidean vectors, as the feet of the quadruped are modeled as *points*. The velocity of the feet relative to the torso origin is the opposite of the velocity of the torso origin relative to the foot. Thus,

$$\mathbf{G}_{f_i} = -\begin{bmatrix} -\mathbf{p}_{f_i}\times & \mathbf{1} \end{bmatrix} \tag{13}$$

where $\mathbf{p}_{f_i}\times \in \mathbb{R}^{3\times 3}$ is the skew-symmetric cross product matrix of the position of the foot in torso coordinates. The vector $\mathbf{g}_{f_i}$ can be directly computed as

$$\begin{aligned} \mathbf{g}_{f_i} &= \dot{\mathbf{G}}_{f_i} \mathbf{v}_t \\ &= -\begin{bmatrix} -\dot{\mathbf{p}}_{f_i}\times & \mathbf{0} \end{bmatrix} \mathbf{v}_t \\ &= \mathbf{v}_{f_i} \times \boldsymbol{\omega}_t \end{aligned} \tag{14}$$

3) Feet to Joint Velocities: In the last stage we have to map feet velocity to joint velocities, finding the matrices $\mathbf{G}_{a_i} \in \mathbb{R}^{3\times 3}$ and $\mathbf{g}_{a_i} \in \mathbb{R}^3$ such that

$$\dot{\mathbf{q}}_{f_i} = \mathbf{G}_{a_i} \mathbf{v}_{f_i} \tag{15a}$$
$$\ddot{\mathbf{q}}_{f_i} = \mathbf{G}_{a_i} \mathbf{a}_{f_i} + \mathbf{g}_{a_i} \tag{15b}$$

with $\mathbf{g}_{a_i} = \dot{\mathbf{G}}_{a_i} \mathbf{v}_{f_i}$. Assuming that each leg of the quadruped has 3 DoFs results in $\mathbf{G}_{a_i} = \mathbf{J}_i^{-1}$, where $\mathbf{J}_i$ is the Jacobian of the stance leg f_i. It then follows that

$$\begin{aligned} \mathbf{g}_{a_i} &= \dot{\mathbf{G}}_{a_i} \mathbf{v}_{f_i} \\ &= \dot{\mathbf{J}}_i^{-1} \mathbf{J}_i \dot{\mathbf{q}}_{f_i} \\ &= \left(-\mathbf{J}_i^{-1} \dot{\mathbf{J}}_i \mathbf{J}_i^{-1}\right) \mathbf{J}_i \dot{\mathbf{q}}_{f_i} \\ &= -\mathbf{J}_i^{-1} \dot{\mathbf{J}}_i \dot{\mathbf{q}}_{f_i} \end{aligned} \tag{16}$$

In this derivation, the formula of the derivative of the inverse of a matrix was used from the second to the third line.

4) Linear combination: The transforms described in (6) are readily constructed by collecting the intermediate terms comprising (7), (12), and (15). For each stance leg f_i, it follows that

$$\dot{\mathbf{q}}_{f_i} = \mathbf{G}_{a_i} \mathbf{G}_{f_i} \mathbf{G}_t \dot{\mathbf{y}} \tag{17}$$

and, similarly,

$$\begin{aligned} \ddot{\mathbf{q}}_{f_i} &= \mathbf{G}_{a_i}\left(\mathbf{G}_{f_i}\left(\mathbf{G}_t \ddot{\mathbf{y}} + \mathbf{g}_t\right) + \mathbf{g}_{f_i}\right) + \mathbf{g}_{a_i} \\ &= \mathbf{G}_{a_i} \mathbf{G}_{f_i} \mathbf{G}_t \ddot{\mathbf{y}} + \mathbf{G}_{a_i} \mathbf{G}_{f_i} \mathbf{g}_t + \mathbf{G}_{a_i} \mathbf{g}_{f_i} + \mathbf{g}_{a_i} \end{aligned} \tag{18}$$

By comparing (17) and (18) with (6), it follows that

$$\mathbf{G}_j = \begin{bmatrix} \mathbf{0} \\ \mathbf{G}_{a_1} \mathbf{G}_{f_1} \mathbf{G}_t \\ \mathbf{G}_{a_2} \mathbf{G}_{f_2} \mathbf{G}_t \\ \mathbf{0} \end{bmatrix} \tag{19}$$

and

$$
\mathbf{g}_j = \begin{bmatrix} \mathbf{0} \\ \mathbf{G}_{a_1} \left(\mathbf{G}_{f_1} \mathbf{g}_t + \mathbf{g}_{f_1} \right) + \mathbf{g}_{a_1} \\ \mathbf{G}_{a_2} \left(\mathbf{G}_{f_2} \mathbf{g}_t + \mathbf{g}_{f_2} \right) + \mathbf{g}_{a_2} \\ \mathbf{0} \end{bmatrix} \quad (20)
$$

for the sample case where the stance legs are the second and third leg of the quadruped. The transforms from the pendulum states to the floating base states are given by (7).

The balance controller uses the states describing the physical process of balancing of the real robot and outputs $\ddot{L}$. Using (19) and (20) we obtain the numeric values of the equivalent equation of motion (EoM) of the virtual model and use these along with $\ddot{L}$ to compute the desired acceleration of the actuated joint, $\ddot{y}_h$, to achieve this output. The controller on the quadruped tries to track the acceleration command required for balancing, but also the configuration that best conforms to the virtual pendulum model. The control law is

$$
\boldsymbol{\tau} = \mathbf{K}_P \left(\mathbf{q}_{vip} - \mathbf{q} \right) + \mathbf{K}_D \left(\dot{\mathbf{q}}_{vip} - \dot{\mathbf{q}} \right) + \mathbf{K}_B \ddot{\mathbf{q}}_{bal} \quad (21)
$$

where $\ddot{\mathbf{q}}_{bal}$ is the desired joint acceleration vector, obtained by applying (6) on the pendulum acceleration commanded by the balance controller. The kinematic mapping (6) also gives us the joint velocity vector $\dot{\mathbf{q}}_{vip}$ corresponding to an accurate mimicking of the pendulum behavior. The position vector $\mathbf{q}_{vip}$ is instead constructed via inverse kinematics:

$$
\mathbf{q}_{f_i,vip} = IK(\mathbf{p}_{f_i}(\mathbf{y})) \quad (22)
$$

where $\mathbf{p}_{f_i} \in \mathbb{R}^3$ is the nominal position of the stance foot when the quadruped is accurately mimicking the inverted pendulum. The vector $\mathbf{q}_{vip}$ is constructed by replacing the joints of the stance legs f_i by $\mathbf{q}_{f_i,vip}$ and leaving the joints of the swing legs unchanged. $\mathbf{K}_P, \mathbf{K}_D, \mathbf{K}_B \in \mathbb{R}^{n \times n}$ are all positive definite and manually tuned based on the desired error dynamics.

V. LOCOMOTION USING THE BALANCE CONTROLLER

So far, we have presented the realization of the pure task of balancing a quadruped by controlling a virtual 2-DoF model associated to the quadruped by the kinematic mapping derived in Sec. IV. Any motion foreign to that generated by the balance controller is effectively seen as a disturbance to the system. Strictly speaking, the operational space of a robot balancing while performing an additional task consists of an operational state used for balancing, and $n-1$ operational states used for the additional motion.

For a system with $n > 2$ DoFs, the control law in [1] still outputs $\ddot{L}$ but is now also dependent on the motion of the additional states. The EoM of the generalized system with the additional fictitious (and static) prismatic joint is

$$
\begin{bmatrix} H_{00} & H_{01} & H_{02} & \mathbf{H}_{03} \\ H_{10} & H_{11} & H_{12} & \mathbf{H}_{13} \\ H_{20} & H_{21} & H_{22} & \mathbf{H}_{23} \\ \mathbf{H}_{30} & \mathbf{H}_{31} & \mathbf{H}_{32} & \mathbf{H}_{33} \end{bmatrix} \begin{bmatrix} 0 \\ \ddot{y}_p \\ \ddot{y}_h \\ \ddot{\mathbf{y}}_m \end{bmatrix} + \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} \tau_0 \\ 0 \\ w_h \\ \mathbf{w}_m \end{bmatrix} \quad (23)
$$

where $\mathbf{y}_m$ contains the actuated generalized coordinates of the motion and the dots indicate time derivatives. In fact,
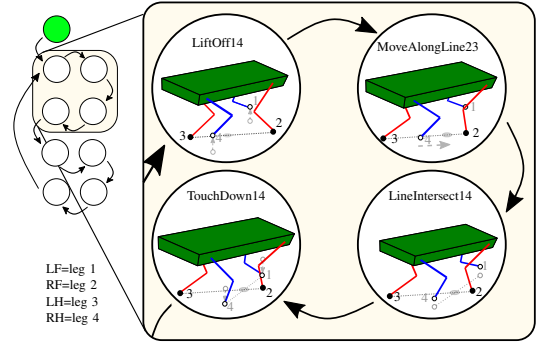

Fig. 4. (Left) Modes of the state machine made to generate the motion of the swing feet during the balanced walk on a line. The FullStance mode (shown in bright green) is executed only at the beginning. The order of the four highlighted modes can be thought of as being repeated in the same order below them but for the LF-RH leg pair. (Right) Detail of the transitions for the RF-LH leg pair.

$\ddot{\mathbf{y}}_m$ is the output of the motion controller. It is shown in [1] (where all parameters are defined) that $\ddot{L}$ and τ_0 are directly related, hence (23) can be separated into known and unknown variables, resulting in the system of equations

$$
\begin{bmatrix} 0 & \mathbf{0} & H_{01} & H_{02} \\ 0 & \mathbf{0} & H_{11} & H_{12} \\ -1 & \mathbf{0} & H_{21} & H_{22} \\ \mathbf{0} & -\mathbf{1} & \mathbf{H}_{31} & \mathbf{H}_{32} \end{bmatrix} \begin{bmatrix} w_h \\ \boldsymbol{w}_m \\ \ddot{y}_p \\ \ddot{y}_h \end{bmatrix} = \begin{bmatrix} -\frac{\ddot{L}}{g} - C_0 - \mathbf{H}_{03} \ddot{\mathbf{y}}_m \\ -C_1 - \mathbf{H}_{13} \ddot{\mathbf{y}}_m \\ -C_2 - \mathbf{H}_{23} \ddot{\mathbf{y}}_m \\ -\mathbf{C}_3 - \mathbf{H}_{33} \ddot{\mathbf{y}}_m \end{bmatrix} \quad (24)
$$

which combines the outputs of the balance and the motion controllers. After solving (24), the generalized forces w_h and $\boldsymbol{w}_m$ are mapped to joint torques through the Jacobian.

For the task of walking on a line while balancing, one can easily design motions that lie on the balance null space. These are motions that do not disturb the CoM of the robot in the direction perpendicular to the line of motion. We chose to generate this motion by dividing it into a series of symmetric motions executed by a state machine, as shown in Fig. 4. The sequence of events was designed as follows. The task is assumed to start with the quadruped having all four legs on the ground (FullStance). The state machine immediately proceeds to the LiftOff14 mode, in which it lifts the LF and RH legs and forces the balance controller to use the RF and LH legs as its stance legs. The robot then balances on the RF-LH leg pair and once the robot attains a balanced state, it proceeds to MoveAlongLine23. The user-specified distance to be traveled by the robot is passed through a PD position controller to regulate the x-component of the feet position, i.e., $\mathbf{p}_{f_i}$ in (22), effectively bringing the robot closer to the front stance leg. Once the robot has reached the desired position, it moves to LineIntersection14 mode, where it positions the swing legs such that the line connecting them intersects the current CoM (x, y)-position while keeping them slightly above the ground. Once this position has been reached, the TouchDown14 event is triggered, which simply brings the swing legs down to touch the ground. The exact same procedure is then repeated for the other leg pair.

The balance controller was integrated into our existing reactive control framework (RCF) [19] mostly to take ad-

vantage of two additional tools: *gravity compensation* and the *kinematic adjustment* to control the position of the feet. The former helps remove most of the undesired terms of the equations of motion, leaving us with the natural task of balancing. The latter provides better accuracy on the positioning of the swing legs during the touchdown event.

VI. Results

In this section we present the results from the tests designed to assess the performance and robustness of the proposed strategy. Our approach is implemented on HyQ [2] and the tests consist of different balancing tasks where the robot uses only its diagonal leg pairs. Three tests are performed in simulation and one experimentally (all the tests can be seen in the accompanying video).

Our control architecture consists of a high-level and a low-level control layer, running at 250 Hz and 1 kHz, respectively. Due to all kinematic transformations involved, we compute the balance torques $\mathbf{K}_B \ddot{\mathbf{q}}_{bal}$ in (21) at the high-level along with the $\mathbf{q}_{vip}$ and $\dot{\mathbf{q}}_{vip}$, The latter two variables are being tracked at low-level for better impedance rendering. The CoM states are computed using only proprioceptive sensors, i.e. joint encoders and an Inertia Measurement Unit, that are fused in a state estimator algorithm [20]. All of the kinematic and dynamic transformations are generated through RobCoGen [21].

A. Simulations

Our goal with the simulations is twofold: understand the performance of the balance controller under ideal conditions and its sensitivity to mis-measurements and to external disturbances. For this, we designed three simulation tests. First, we assess the controller response when HyQ is balancing on two diagonal legs and experiences a constant external disturbance. We chose a constant disturbance as a way of understanding the side effect of problems like torque sensor offset/miscalibration, inaccuracy in kinematic parameters and/or overall robot CoM position, and also external disturbances from the hydraulic hoses that are connected to HyQ. In a second test we evaluate the controller performance when the robot is required to balance and track a desired trunk trajectory at the same time. The third test concludes the simulations with a task we called *The Ninja Walk*, where the algorithm proposed in Sec. V is evaluated in an extreme locomotion scenario.

1) Constant disturbance: in this test, the robot starts from a full stance condition and proceeds to balance on two legs by simply raising a pair of diagonal legs. When HyQ reaches a steady condition, a constant force is applied on its floating base, in a direction perpendicular to the line of contact formed by the stance legs (the most sensitive orientation for the balance controller). The force is applied for about 10 seconds until it reaches a new steady state. This process is then repeated with increasing forces until the robot destabilizes or touches the ground with its non-supporting legs. The results are shown in Fig. 5. It can be seen that the steady state of the virtual joint y_h deviates from
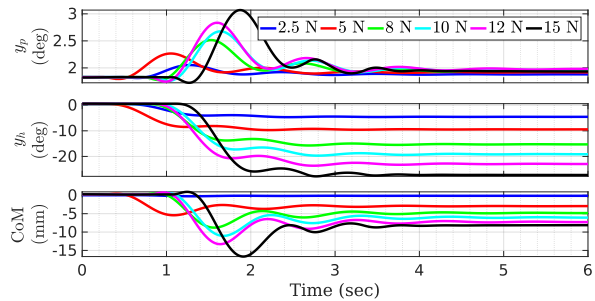

Fig. 5. Trajectories of the equivalent (top) pivot joint and (middle) hip joint of the virtual model after HyQ is pushed on the trunk with a constant force, starting from a balanced condition. (bottom) The corresponding distance of the CoM of HyQ w.r.t. the contact line, i.e., along the `vbase` frame.

the desired zero value as the magnitude of the disturbance force increases. This motion moves the CoM away from the pivot point to compensate for the moment created by the disturbance. At 15 N of applied force, the robot starts touching the ground with one of its non-supporting legs.

2) Tracking a reference trajectory: as mentioned in Sec. III, knowledge of the reference trajectory allows us to compensate for the inherent non-minimum phase behavior of the balancing process. To do so, the trajectory has to be recomputed by passing it through a low-pass filter running backwards in time, as explained in [1]. The tracking task regards only the second joint of the virtual model y_h. Trying to understand the balance controller response to different input signals, we chose a reference trajectory composed of a series of ramps with fast and slow slopes, and a sine wave, as shown in Fig. 6. For this test, HyQ assumes a balancing configuration where the torso (`rtorso` frame) is located 0.47 m from the ground (lower than the one used for the disturbance test), and with the non-supporting legs wide open. This results in a higher velocity gain that, according to the theory, leads to a better balancing performance. The wide open posture for the non-supporting legs is chosen to prevent them from coming into contact with the ground during the motion of the torso.

The tracking performance and control inputs are shown in Fig. 6. The reference trajectory is tracked with a maximum overshoot of 6%, at the end of the steep ramps, and a milder 2% overshoot at the end of the slower ramps. *Leaning in anticipation* can be seen by the motion of the virtual pivot joint y_p (prior to each of the ramps). This moves in the same direction of the ramp slopes. It is important to notice that, even considering a robot model of 90 kg, the torques required from the controller to balance tend to be quite small. During steady states, most of the joint efforts are carried by the gravity compensation torques.

3) Walking across a narrow bridge (The Ninja Walk): we have considered a narrow bridge as an extreme locomotion scenario where footstep locations are very limited, offering very small contact regions and creating support polygons with almost no area; in other words, a condition where a robot would only be able to cross if balancing on one or two
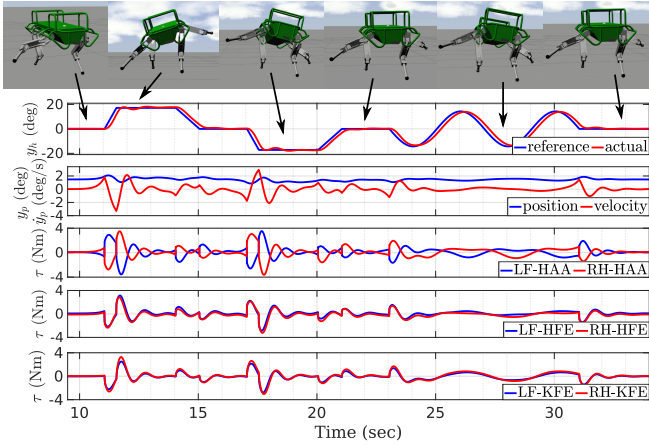
Fig. 6. (top) Frames of the simulation of HyQ tracking a reference trajectory (second row) in which the reference angle is the angle about the axis formed by the line connecting the stance legs. (third row) *Leaning in anticipation* caused by the filtered signal. (fourth to sixth rows) Torques of the joints of the stance legs as computed by the control law (21).
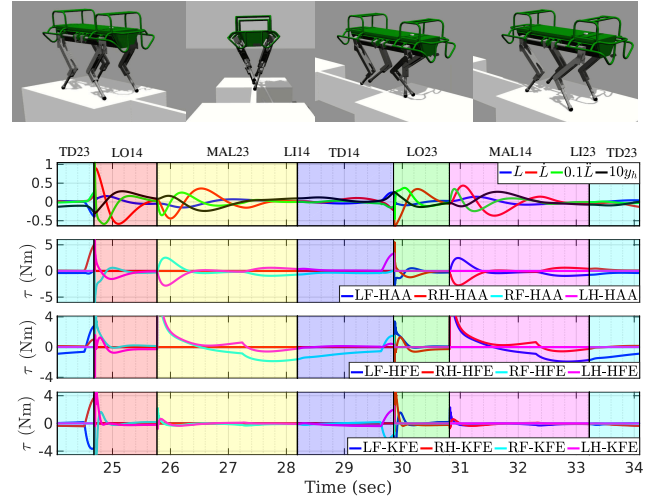

Fig. 7. (top) Snapshots of HyQ performing the *Ninja Walk* on a narrow bridge. (bottom) Zoomed-in plots showing the states of the plant converging to zero before the robot starts to move along the line. The shaded regions correspond to different modes of the state machine described in Sec. V.

legs and without depending on contact areas. The narrow bridge is 1.5 m long and 6 cm wide, as depicted in Fig. 1, and has the same stiffness and contact properties as the ground in the previous simulations. For comparison, the point-feet of HyQ are essentially spheres with a diameter of 4 cm.

The motion results of the *Ninja Walk* are detailed in Fig. 7. HyQ successfully crosses the bridge in about 2 minutes, counting from the initial to the final full-stance configurations (as shown in the left and right-most snapshots of Fig. 7). The many graphics comprise a close-up of a full cycle of the state machine described in Sec. V, which lasts roughly 10 seconds. The various states of the state machine are identified with different colors, with their associated names abbreviated on the top of the graph. For instance, the first state, abbreviated as *TD23*, corresponds to the TouchDown23 state from Fig. 4 and is followed by LiftOff14, then MoveAlongLine23, and so on. The transition events that disturb the robot the most are LO14, LO23, MAL14 and MAL23, as seen by the largest variations in $\dot{L}$. Note that the lift-off events additionally incur a slightly discontinuous jump on the position of the virtual pendulum states as the model switches from one leg-pair to the other. Note also that towards the end of the touch-down events, a slightly de-stabilizing motion can be seen. This is due to the motion of the swing legs that is generated to track a landing position in the world frame, whose control actions are not in agreement with the balance controller.

B. Experimental validation

In this section we show the preliminary results and outcomes from the implementation of the proposed approach on a real legged machine. The experiment consists of starting the robot in a three-leg stance condition and then manually bringing its torso towards its balanced position, with the balance controller activated. After HyQ balances on its two legs, we perturb it by pushing and pulling it from its protection frame to bring it out of balance a few times. Next, it is left alone to regain balance. This simple experiment

serves to understand the performance and sensitivity of the balance controller given the limitations of the real system described in the beginning of Sec. VI-A.

The most relevant experimental data is shown in Fig. 8. The colored pictures on the top are snapshots from the experiment that have their corresponding signals in the bottom plots. In the translucent yellow area, it can be seen that when the robot is moved from three-leg to two-leg stance, the distance from the CoM to the contact line (fifth plot from top to bottom) gets closer to zero and remains in the vicinity as it balances alone. During a balancing steady-state, none of the quantities converge to a fixed value. This corresponds to the natural process of balancing, since the only way of achieving stability in an inherently unstable system is keeping it in motion. Note, however, that the quantities L, $\dot{L}$ and the CoM distance to the support line do not oscillate around zero. This behavior resembles the steady-state conditions observed in the constant disturbance test, shown in Sec. VI-A.1, where the disturbance in this case are mainly related to: the inaccuracy of torque sensors; a wrong estimation of the CoM position; and the external forces and moments from the hydraulic hoses connected to HyQ. The latter was clearly observed during the experiments. The mechanical flexibility of the robot structure is part of the non-modeled dynamics that constitutes a relevant source of disturbances that substantially affects the estimation of the CoM related states.

The translucent green area in Fig. 8 corresponds to the manually applied impulsive disturbances, which can be seen from the big variations in y_h, which is closely related to the motion of the torso. After the impulsive disturbances, the robot comes back to another balancing steady-state (translucent blue area).

VII. CONCLUSIONS

This work proposes a balancing approach that does not rely on support polygons and can be applied to legged

the non-supporting legs.

REFERENCES

[1] R. Featherstone, "A simple model of balancing in the plane and a simple preview balance controller," *The Int. Journal of Robotics Research (IJRR)*, vol. 36, no. 13-14, pp. 1489–1507, 2017.

[2] 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*, vol. 225, no. 6, pp. 831–849, 2011.

[3] M. Gajamohan, M. Merz, I. Thommen, and R. D'Andrea, "The Cubli: A cube that can jump up and balance," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 3722–3727.

[4] P. Y. Lam, "Gyroscopic stabilization of a kid-size bicycle," *IEEE Int. Conf. on Cybernetics and Intell. Systems (CIS)*, pp. 247–252, 2011.

[5] U. Nagarajan, G. Kantor, and R. Hollis, "The ballbot: An omnidirectional balancing mobile robot," *The Int. Journal of Robotics Research (IJRR)*, vol. 33, no. 6, pp. 917–930, 2014.

[6] J. J. M. Driessen, A. E. Gkikakis, R. Featherstone, and B. R. P. Singh, "Experimental Demonstration of High-Performance Robotic Balancing," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 9459–9465.

[7] B. Stephens, "Integral control of humanoid balance," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2007, pp. 4020–4027.

[8] B. J. Stephens and C. G. Atkeson, "Push recovery by stepping for humanoid robots with force controlled joints," *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, pp. 52–59, 2010.

[9] T. Koolen, T. de Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models," *The Int. Journal of Robotics Research (IJRR)*, vol. 31, no. 9, pp. 1094–1113, 2012.

[10] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Resolved Momentum Control: Humanoid Motion Planning Based on the Linear and Angular Momentum," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, pp. 1644–1650, 2003.

[11] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, 2016.

[12] B. Henze, A. Dietrich, and C. Ott, "An Approach to Combine Balancing with Hierarchical Whole-Body Control for Legged Humanoid Robots," *IEEE Robotics and Automation Letters (RAL)*, vol. 1, no. 2, pp. 700–707, 2016.

[13] G. Wiedebach, S. Bertrand, T. Wu, L. Fiorio, S. McCrory, R. Griffin, F. Nori, and J. Pratt, "Walking on partial footholds including line contacts with the humanoid robot atlas," in *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 1312–1319.

[14] B. H. Kim, "Centroid-based analysis of quadruped-robot walking balance," *Int. Conf. on Advanced Robotics (ICAR)*, pp. 1–6, 2009.

[15] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain," *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 2665–2670, 2010.

[16] M. Chignoli and P. M. Wensing, "Variational-Based Optimal Control of Underactuated Balancing for Dynamic Quadrupeds," *IEEE Access*, vol. 8, pp. 49 785–49 797, 2020.

[17] R. Featherstone, "Quantitative measures of a robot's physical ability to balance," *The Int. Journal of Robotics Research (IJRR)*, vol. 35, no. 14, pp. 1681–1696, 2016.

[18] ——, *Rigid Body Dynamics Algorithms*. New York, NY: Springer, 2008.

[19] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 2554–2561.

[20] S. Nobili, M. Camurri, V. Barasuol, M. Focchi, D. G. Caldwell, C. Semini, and M. Fallon, "Heterogeneous sensor fusion for accurate state estimation of dynamic legged robots," in *Robotics: Science and Systems (RSS)*, Boston, USA, July 2017.

[21] 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 (JOSER)*, vol. 7, no. July, pp. 36–54, 2016.
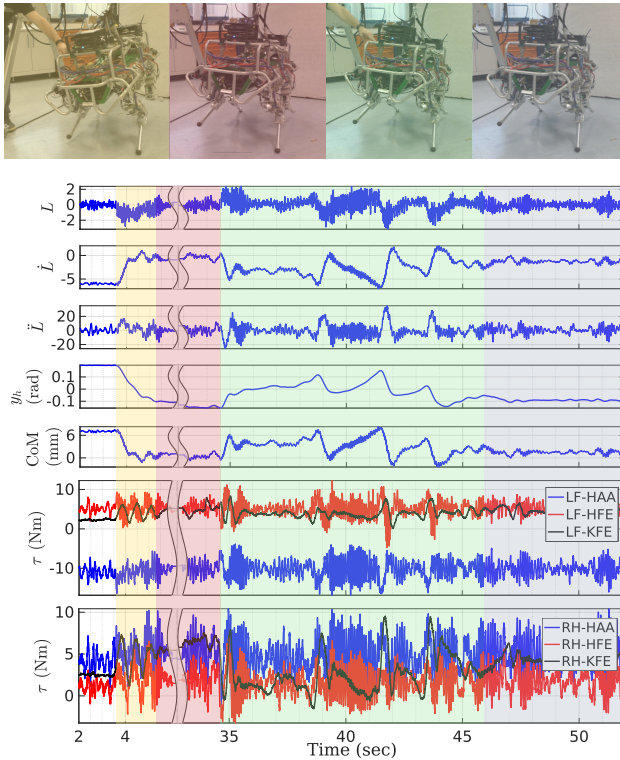
Fig. 8. Snapshots and associated data of the balancing experimental test performed on HyQ: the robot is moved from three-leg to two-leg stance (yellow area); the red area represents balancing steady-state conditions before applying disturbances to the robot torso (repeated data is shrunk); series of impulsive disturbances are manually applied upward and downward to the robot torso (green area). Graphics on the bottom are states of the plant in Fig. 2 and the torques computed according to (21). (Plot made with BreakXAxis function created by J. Haas)

robots that have flat-feet or even point-feet. The approach is based on the balancing method presented in [1] used to balance a 2-DoF virtual model, and a kinematic mapping to transform the virtual model quantities into the robot joint space. Furthermore, we showed that a rather simple motion generation algorithm combined with the proposed balancing approach would allow a robot to walk over extremely difficult scenarios. Simulation results were designed to assess the controller performance and to understand its sensitivity to disturbances that can be encountered in a real implementation. We presented preliminary experimental results showing for the first time a point-feet quadruped robot balancing on two legs. Both simulation and experimental tests were very useful to bring knowledge on the requirements to achieve good balancing performance. The most relevant findings are: a) it is important having accurate torque measurements (for torque-controlled robots), since the balance controller generates low torque commands; the reference tracking task is very sensitive to disturbances, that can be represented by torque offsets, external forces or biased CoM estimation; and any non-modelled flexibility along the robot kinematic structure can introduce unrealistic CoM measurements.

Future work considers improving the balance performance by extending the virtual model to also output commands to