Zero Step Capturability for Legged Robots in Multi Contact

Andrea del Prete, Steve Tonneau, Nicolas Mansard

Zero Step Capturability for Legged Robots in Multi Contact

Andrea Del Prete, *Member, IEEE,* Steve Tonneau and Nicolas Mansard

***Abstract*—The ability to anticipate a fall is fundamental for any robot that has to balance. Currently, fast fall-prediction algorithms only exist for simple models, such as the Linear Inverted Pendulum Model (LIPM), whose validity breaks down in multi-contact scenarios (i.e. when contacts are not limited to a flat ground). This paper presents a fast fall-prediction algorithm based on the point-mass model, which remains valid in multi-contact scenarios. The key assumption of our algorithm is that, in order to come to a stop without changing its contacts, a robot only needs to accelerate its center of mass in the direction opposite to its velocity. This assumption allows us to predict the fall by means of a convex optimal control problem, which we solve with a fast custom algorithm (less than 11 ms of computation time). We validated the approach through extensive simulations with the humanoid robot HRP-2 in randomly-sampled scenarios. Comparisons with standard LIPM-based methods demonstrate the superiority of our algorithm in predicting the fall of the robot, when controlled with a state-of-the-art balance controller. This work lays the foundations for the solution of the challenging problem of push recovery in multi-contact scenarios.**

***Index Terms*—Stability, Viability, Legged Robots, Multi-Contact.**

I. Introduction

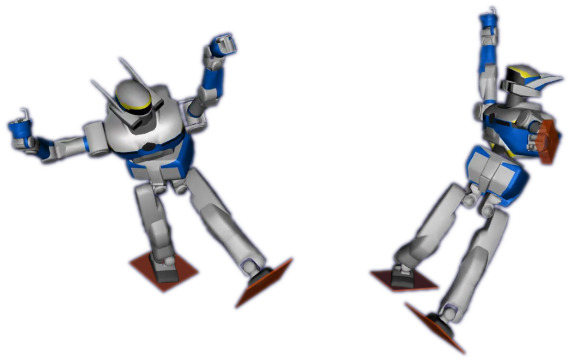
A. Overview

The main issue preventing legged robots from being deployed outside research laboratories is probably their current unsafeness. Their unstable dynamics makes balancing these systems a real challenge. It is thus crucial to endow legged robots with the ability to avoid falling, or at least to fall in such a way that minimizes damage to people and objects in the surroundings. A prerequisite to tackle this issue is of course the ability to quickly predict the fall, known in robotics as the problem of capturability [1], [2]. Many tools for fall prediction (such as the capture point [3]) have been presented, but they only apply to specific situations (e.g. level ground). For the general case (such as the scenarios depicted in Fig. 1), no solution exists that is fast enough to be of any practical use. Since the whole point of equipping robots with legs is to allow them to locomote on irregular grounds, restricting their application to quasi-flat level ground appears as a severe shortcoming [4].

This paper presents a fast and general fall-prediction algorithm for legged robots in multi-contact situations, and assesses its accuracy through simulations with the humanoid robot HRP-2. Our algorithm can also estimate how close the system is to falling, which provides useful insight and could be used for controller design [1].

The authors are with the CNRS, LAAS, 7 avenue du colonel Roche, Univ de Toulouse, LAAS, F-31400 Toulouse, France. e-mail: adelpret@laas.fr, nmansard@laas.fr.

Manuscript received ...



(a) Two (non-coplanar) contacts. (b) Three contacts.

Fig. 1. Examples of the humanoid robot HRP-2 falling in multi-contact simulation scenarios.

The problem of fall prediction is very much related to the ones of balancing, push-recovery and fall-damage minimization. In general, all of these problems are intractable for the complex model of articulated robots, which motivated the use of the *Linear Inverted Pendulum Model* (LIPM). This simple linear model turns out to be a reasonable approximation of a legged robot as long as these hypotheses are satisfied (for more details see [5], Section 48.2.2):

1) the contact points lie on the same plane [6];
2) the center of mass (CoM) of the robot moves on a plane parallel to the one of the contact points;
3) the angular momentum of the robot is constant (typically zero);
4) friction is sufficiently high to avoid slippage.

This has allowed researchers to devise simple and effective solutions to the above-mentioned problems for the case of level ground. However, these algorithms do not scale to the more general multi-contact case (i.e. when contact points are not coplanar).

Another common reduced model is the point-mass model (introduced in Section II), which only assumes constant angular momentum, so it can be used in multi-contact scenarios. However, this model is nonconvex, which makes the associated algorithms too slow for real applications [7], [8].

Our method is based on the simplifying assumption that, in order to come to a stop, a robot only needs to accelerate its CoM in the direction opposite to its current velocity. We show in Section III that this assumption makes the capturability problem convex, and we can thus propose a fast algorithm to solve it. We then propose in Section IV a simple extension of the capture point, which drastically improves its performance

in multi-contact scenarios—even though our tests (presented in Section V) show that it remains inferior to our algorithm.

B. State of the Art

The problem of fall prediction is closely related to the concept of *viability kernel* [9], defined as the set of all the states from which the legged system can avoid falling. By definition, as soon as the system state leaves the viability kernel, the robot is going to fall. Unfortunately, for complex nonlinear systems such as legged robots, computing the viability kernel seems computationally intractable. A slightly simpler condition is the *N-step capturability* [1], [2], which is the ability of a legged system to come to a stop without falling by taking at most N steps. Capturability has been thoroughly studied for the LIPM (or slight extensions of it), which allows for an analytical computation of the *capture point*: the point on the ground where to step to come to a stop [3].

Several extensions of the capture point have been proposed to overcome its limitations (i.e. the above-mentioned LIPM hypotheses). For instance, the Generalized Foot Placement Estimator (GFPE) [10] takes into account a non-level ground with discrete slope changes. Another extension of the capture point to quadratic CoM paths (with varying CoM height) and a polygonal representation of the terrain has been proposed in [11]. The Divergent Component of Motion [12] has been proposed as a 3D extension of the capture point to plan and control bipedal locomotion over rough terrains. These works extend the capture point to more general cases, but none of them address the multi-contact scenario (e.g. by considering contacts with a vertical surface). A pragmatic approach to use the capture point in multi-contact scenarios is to introduce a CoM offset in the LIPM dynamics, and estimate it with a Kalman filter [13]. Even though this method has been used on a real robot, no theoretical analysis justifies its soundness.

In [14] the authors proposed to use machine learning to predict humanoid fall. Another machine-learning approach to instability detection of bipedal robots was presented in [15], with a final reaction time of about 60 ms. In [16], instability is detected by monitoring the deviations of the attitude from a reference model. The main strength of machine-learning based approaches is their capability to capture complex constraints, which may be impossible to describe with any reasonable model. However, their main limitation is the lack of guarantees of generalization outside the training dataset. Moreover, in certain cases, computation times may be too slow for push recovery.

Several researchers have also dealt with the problem of fall-damage minimization. In [17] the authors proposed a fall controller that changes the fall direction to avoid hitting people or objects in the surroundings. An optimal planning of falling motions for humanoid robots to reduce the damage has been investigated in [18]. In [19] the authors presented an optimization-based control strategy to generate whole-body trajectories to minimize fall damage. Given an unstable initial state of the robot, Liu et al. [20], [21] found the optimal contact sequence to dissipate the initial momentum with minimal impacts on the robot. These fall-damage minimization algorithms could be used in combination with our algorithm, in case a fall is predicted and balance seems impossible to recover.

The approach that is the closest to ours is the optimization-based push recovery for multi-contact scenarios [7], [8]. This method is based on our same reduced model (i.e. a point-mass) and it presents a *dynamic stability indicator* that resembles our capturability criterion. Its main limitation is that it needs to solve several nonconvex discretized optimal-control problems, which makes it too slow for real-time applications (about 0.7 s) and subject to local minima.

C. Contributions

We list here the main contributions of this work.

- We propose the first fast (<11 ms) algorithm for fall prediction of legged robots in multi-contact scenarios.
- We empirically demonstrate the good fall-prediction capabilities of our algorithm through thousands of simulations with randomly-sampled initial conditions.
- We empirically evaluate the fall-prediction capabilities of the capture point, showing that it performs poorly in multi-contact scenarios.
- We propose a simple extension of the capture point (by checking its membership to the support polygon [22]), and we empirically show that it is a reasonable fall indicator in multi-contact scenarios—although not as good as our method.

II. Definitions and Problem Statement

A. Centroidal Dynamics

Considering a robot in contact with the environment at k contact points, its Newton-Euler equations are:

$$m\,\ddot{\mathbf{c}} = \sum_{i=1}^{k} \mathbf{f}_i + m\mathbf{g} \tag{1a}$$

$$\dot{\mathbf{l}} = \sum_{i=1}^{k} (\mathbf{p}_i - \mathbf{c}) \times \mathbf{f}_i \tag{1b}$$

where $m \in \mathbb{R}$ is the robot mass, $\mathbf{c} \in \mathbb{R}^3$ is the CoM position, $\mathbf{f}_i \in \mathbb{R}^3$ is the i-th contact force, $\mathbf{g} = [0, 0, -9.81]^\top$ is the gravity acceleration, $\mathbf{l} \in \mathbb{R}^3$ is the angular momentum (expressed at the CoM) and $\mathbf{p}_i \in \mathbb{R}^3$ is the i-th contact point. All quantities are expressed in an arbitrary inertial frame having $\mathbf{z}$ aligned with the gravity. Equation (1b) can be reformulated as:

$$\dot{\mathbf{l}} + \mathbf{c} \times \left(\sum_{i=1}^{k} \mathbf{f}_i \right) = \sum_{i=1}^{k} \mathbf{p}_i \times \mathbf{f}_i \tag{2}$$

By replacing $\sum_{i=1}^{k} \mathbf{f}_i$ with $m(\ddot{\mathbf{c}} - \mathbf{g})$, we reformulate (1) as:

$$\underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\,\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{l}} \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \dots & \hat{\mathbf{p}}_k \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_k \end{bmatrix}}_{\mathbf{f}}, \tag{3}$$

where $\hat{\mathbf{p}} \in \mathbb{R}^{3\times3}$ is the cross-product matrix related to $\mathbf{p}$. We call $\mathbf{w} = \mathbf{A}\mathbf{f} \in \mathbb{R}^6$ the centroidal wrench (also called pseudo-wrench [23]).

B. Centroidal Cone

According to Coulomb's law, each contact force is constrained to lie inside a friction cone:

$$||\mathbf{f}_i - (\mathbf{n}_i^\top \mathbf{f}_i)\mathbf{n}_i|| \leq \mu_i \mathbf{n}_i^\top \mathbf{f}_i \qquad \forall i = 1 \dots k, \tag{4}$$

where μ_i is the friction coefficient, and $\mathbf{n}_i \in \mathbb{R}^3$ is the normal direction at the i-th contact. A very common alternative to the quadratic friction-cone constraints (4) is to approximate them with polytopes [7], [22], [23]. We can express the linearized friction-cone constraints as a set of linear inequalities:

$$\mathbf{B}\,\mathbf{f} \leq \mathbf{0} \tag{5}$$

Equations (3) and (5) imply that the set of admissible centroidal wrenches $\mathbf{w}$ is also a cone. Its linearization can be computed using polytope-projection techniques [24]. We represent this *centroidal cone* with a matrix $\mathbf{H}$ such that:

$$\mathbf{H}\mathbf{w} \leq \mathbf{0} \iff \exists \mathbf{f} : \mathbf{B}\mathbf{f} \leq \mathbf{0}, \mathbf{w} = \mathbf{A}\,\mathbf{f} \tag{6}$$

C. Problem Statement

We consider the *0-step capturability problem*, which consists in determining whether the system can come to a stop without moving the current contact points. Solving this problem for a given state considering the full dynamics of a legged robot is too computationally expensive to be of any practical use. Instead, we take the common approach of considering only the centroidal dynamics of the system (1), which greatly reduces the size of the problem.

The main concern when using the centroidal dynamics is given by the angular momentum $\mathbf{l}$. On the one hand we know that angular momentum is a great resource for balancing, so we would like to exploit it in our reduced models. On the other hand, the angular momentum is bounded by the limited rotational capabilities of the robot bodies (i.e. joint position and velocity limits), which do not appear in the centroidal model. For this reason, we take the common assumption that $\dot{\mathbf{l}} = 0$ [7], [9], which leads us to a point-mass model.

The 0-step capturability problem can be formulated as a minimum-time optimal control problem:

$$\begin{aligned} \underset{\mathbf{c}(t),\dot{\mathbf{c}}(t),\ddot{\mathbf{c}}(t),T}{\text{minimize}} \quad & T \\ \text{subject to} \quad & \frac{d}{dt}\begin{bmatrix}\mathbf{c}(t)\\ \dot{\mathbf{c}}(t)\end{bmatrix} = \begin{bmatrix}\dot{\mathbf{c}}(t)\\ \ddot{\mathbf{c}}(t)\end{bmatrix} \qquad \forall t \geq 0 \\ & \mathbf{H}\mathbf{w}(\mathbf{c}(t), \ddot{\mathbf{c}}(t)) \leq 0 \qquad \forall t \geq 0 \\ & (\mathbf{c}(0), \dot{\mathbf{c}}(0)) \quad \text{fixed} \\ & \dot{\mathbf{c}}(t) = 0 \qquad\qquad \forall t \geq T, \end{aligned} \tag{7}$$

where $\mathbf{w}$ is seen as a function of $\mathbf{c}$ and $\ddot{\mathbf{c}}$ according to (3). If the solution $T \in \mathbb{R}^+$ is a finite number, then the state is capturable (in the following we omit the prefix "0-step" when we talk about capturability). Even if minimizing the time is not necessary to determine whether a state is capturable, it is useful to compute the so-called *capturability margin* (see Section III-D for more details). The main difficulty in solving (7) comes from the centroidal-cone constraints. These constraints are indeed bilinear because of the cross-product between $\mathbf{c}$ and $\ddot{\mathbf{c}}$, which makes (7) nonconvex.

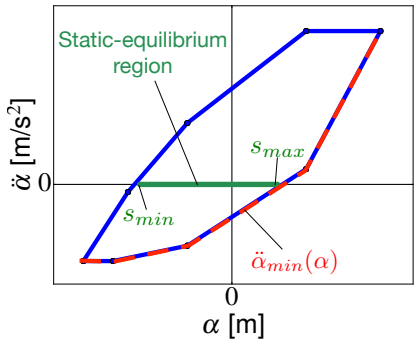


Fig. 2. Typical polytope of feasible CoM accelerations (parametrized as $\ddot{\alpha}$) as a function of CoM positions (parametrized as α). This polytope clearly depicts the unstable dynamics of the system: if $\alpha > s_{max}$ ($\alpha < s_{min}$) and $\dot{\alpha}$ is positive (negative), then the system can no longer come to a stop.

III. Problem Solution

A. Parametrized Centroidal Dynamics

Our approach is based on a simplifying assumption that makes problem (7) convex. We assume that the best strategy to stop the CoM consists in accelerating it in the direction opposite to its velocity. This implies that $\dot{\mathbf{c}}$ and $\ddot{\mathbf{c}}$ are always parallel (which is also what happens when stepping on the capture point). To take advantage of this assumption, we rewrite $\mathbf{c} \times \ddot{\mathbf{c}}$ as:

$$\mathbf{c}(t) \times \ddot{\mathbf{c}}(t) = (\mathbf{c}(0) + \Delta\,\mathbf{c}(t)) \times \ddot{\mathbf{c}}(t), \tag{8}$$

where $\Delta\,\mathbf{c}(t) = \mathbf{c}(t) - \mathbf{c}(0)$ is our new position variable. Since $\Delta\,\mathbf{c}(t)$ and $\ddot{\mathbf{c}}(t)$ are always parallel, $\Delta\,\mathbf{c}(t) \times \ddot{\mathbf{c}}(t) = \mathbf{0}$, so the centroidal-cone constraints become linear:

$$\mathbf{H}\left(\begin{bmatrix} m\mathbf{I}_3 \\ m\hat{\mathbf{c}}(0)\end{bmatrix}\ddot{\mathbf{c}}(t) + \begin{bmatrix}\mathbf{0}\\ m\hat{\mathbf{g}}\end{bmatrix}\Delta\,\mathbf{c}(t) + \begin{bmatrix}-m\mathbf{g}\\ m\mathbf{g}\times\mathbf{c}(0)\end{bmatrix}\right) \leq \mathbf{0} \tag{9}$$

Besides making (7) convex, this assumption reduces the size of the problem. Since $\ddot{\mathbf{c}}(t)$ and $\dot{\mathbf{c}}(t)$ are parallel, the CoM moves on a straight line, so we can parametrize its 3d trajectory by means of a 1d trajectory $\alpha(t)$:

$$\begin{aligned} \mathbf{c}(t) &= \mathbf{c}(0) + \alpha(t)\mathbf{v} \\ \dot{\mathbf{c}}(t) &= \dot{\alpha}(t)\mathbf{v} \\ \ddot{\mathbf{c}}(t) &= \ddot{\alpha}(t)\mathbf{v}, \end{aligned} \tag{10}$$

where $\mathbf{v} \triangleq \frac{\dot{\mathbf{c}}(0)}{||\dot{\mathbf{c}}(0)||}$. Thanks to this parametrization, the centroidal-cone constraints become:

$$\underbrace{\mathbf{H}\begin{bmatrix} m\mathbf{I}_3 \\ m\,\mathbf{c}(0)\times\mathbf{v}\end{bmatrix}}_{\mathbf{a}}\ddot{\alpha}(t) + \underbrace{\mathbf{H}\begin{bmatrix}\mathbf{0}\\ m\mathbf{g}\times\mathbf{v}\end{bmatrix}}_{\mathbf{b}}\alpha(t) \leq \underbrace{\mathbf{H}\begin{bmatrix} m\mathbf{g}\\ m\,\mathbf{c}(0)\times\mathbf{g}\end{bmatrix}}_{\mathbf{d}} \tag{11}$$

This set of inequalities defines the polytope of feasible CoM position-acceleration pairs $(\alpha, \ddot{\alpha})$ along the direction $\mathbf{v}$ (see for instance Fig. 2). Starting from $\alpha(0) = 0$ we can then search for a feasible acceleration trajectory $\ddot{\alpha}(t)$ that allows the system to stop (if any exists). Problem (7) then becomes:

$$\begin{aligned} \underset{\alpha(t),\dot{\alpha}(t),\ddot{\alpha}(t),T}{\text{minimize}} \quad & T \\ \text{subject to} \quad & \frac{d}{dt}\begin{bmatrix}\alpha(t)\\ \dot{\alpha}(t)\end{bmatrix} = \begin{bmatrix}\dot{\alpha}(t)\\ \ddot{\alpha}(t)\end{bmatrix} \quad \forall t \geq 0 \\ & \mathbf{a}\,\ddot{\alpha}(t) + \mathbf{b}\alpha(t) \leq \mathbf{d} \quad \forall t \geq 0 \\ & (\alpha(0), \dot{\alpha}(0)) \quad \text{fixed} \\ & \dot{\alpha}(t) = 0 \quad \forall t \geq T \end{aligned} \tag{12}$$

This new 1d optimal-control problem shares many features with the Time Optimal Path Parametrization (TOPP) problem [25]. However, in our problem the path is not fixed: we know that it will be on a straight line, but we do not know how far the CoM will travel. Moreover, our inequality constraints are affine, which allows to use closed-form solutions to integrate our dynamical system. We propose thus in the following a dedicated algorithm to solve this problem.

B. Algorithm Overview

The key idea of the algorithm is to integrate the linear dynamical system (LDS) $(\alpha, \dot{\alpha})$ applying the maximum deceleration (i.e. minimum acceleration), until either we reach $\dot{\alpha} = 0$ (and we can maintain it), or we can prove that we will never reach it. In addition to the following explanation, we refer the reader to the companion video, which provides a graphical description of the algorithm. Since the space of feasible $(\alpha, \ddot{\alpha})$ is a polytope, the minimum feasible acceleration, denoted by $\ddot{\alpha}_{min}$, is a piecewise-linear function of α (e.g. see Fig. 2). The same applies to the maximum feasible acceleration, $\ddot{\alpha}_{max}$. This means that we can express $\ddot{\alpha}_{min}$ as:

$$\ddot{\alpha}_{min}(\alpha) = \beta_i + \gamma_i \alpha, \qquad \underline{\alpha}_i \leq \alpha \leq \bar{\alpha}_i, \quad i \in [1, N], \tag{13}$$

where N is the number of linear intervals of $\ddot{\alpha}_{min}$. With an abuse of notation, in the following we denote with $\gamma(\alpha)$ the value of γ_i corresponding to the interval $[\underline{\alpha}_i, \bar{\alpha}_i]$ containing α.

Starting from the initial state, we set $\ddot{\alpha}(t) = \ddot{\alpha}_{min}(\alpha(t))$ and integrate the system until either of these conditions is met:

- **C1** $\ddot{\alpha}_{min}(\alpha(t)) \geq 0$ and $\gamma(\alpha(t)) \geq 0$;
- **C2** there is no feasible $\ddot{\alpha}$ for $\alpha(t)$;
- **C3** $\dot{\alpha}(t) = 0$.

In **C1** we know that the system will diverge because it will no longer be able to decelerate (e.g. Fig. 2). In **C2** we reached the right extremity of the $(\alpha, \ddot{\alpha})$ polytope, meaning that there exists no CoM acceleration that allows the robot to maintain the current contacts. In **C3** the system came to a stop at time $t = t_{zv}$. The final answer depends then on the location of $\alpha(t_{zv})$ with respect to the static-equilibrium region $\mathcal{S} : [s_{min}, s_{max}] \subset \mathbb{R}$.

C3.1: If $\alpha(t_{zv})$ belongs to $\mathcal{S}$ (i.e. $s_{min} < \alpha(t_{zv}) < s_{max}$) then the system can maintain $\dot{\alpha}$ at zero because $\ddot{\alpha} = 0$ is feasible. The initial state is then capturable.

C3.2: If $\alpha(t_{zv})$ is located before $\mathcal{S}$ (i.e. $\alpha(t_{zv}) < s_{min}$), it means that we decelerated too quickly to reach $\mathcal{S}$. We thus go back to the initial state and apply $\ddot{\alpha}_{max}$ (which is negative) until either of these conditions is met:

- **C3.2.1** $\ddot{\alpha}_{max}(\alpha(t)) = 0$;
- **C3.2.2** $\dot{\alpha}(t) = 0$.

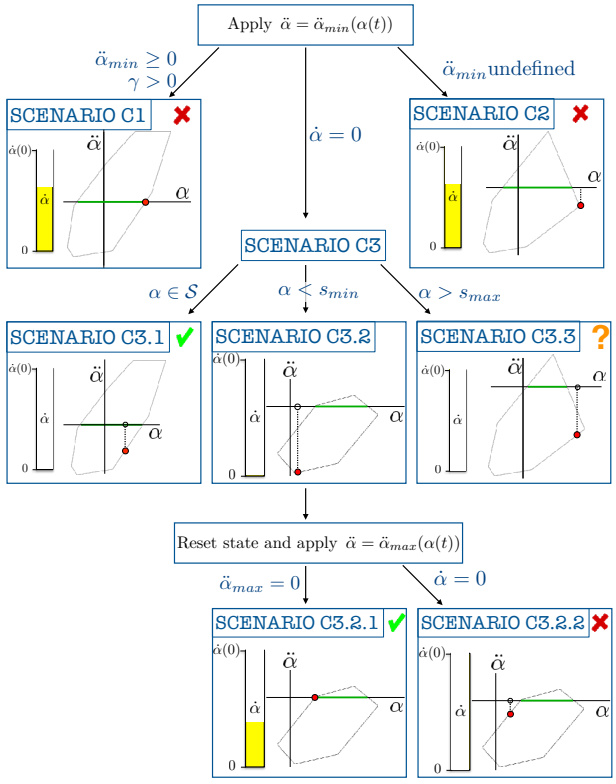


Fig. 3. Decision tree of the capturability algorithm. The yellow bar on the left side of the plot represents the residual velocity $\dot{\alpha}$.

In **C3.2.1** the system reached $\mathcal{S}$, so the initial state was capturable. We do not need to prove that the system can stop inside $\mathcal{S}$ because we have already shown that $\ddot{\alpha}_{max}$ leads the system to $\mathcal{S}$ with a positive velocity, and $\ddot{\alpha}_{min}$ stops the system before reaching $\mathcal{S}$. There must exists then a convex combination of these two trajectories that leads the system to $\mathcal{S}$ with zero velocity.

In **C3.2.2** we know that, despite applying the maximum acceleration, we did not reach $\mathcal{S}$. This means that $\mathcal{S}$ is not reachable from the given initial state, which thus is not capturable.

C3.3: If $\alpha(t_{zv})$ is located after $\mathcal{S}$ (i.e. $\alpha(t_{zv}) > s_{max}$), it means that the CoM dynamics is naturally stable: regardless of the choice of $\ddot{\alpha}$, the system can hardly diverge. This can happen only for very unusual contact geometries, such as if the CoM is located below the contact points (e.g. the robot is hanging from above). A simple way to deal with this case would be to restart the algorithm inverting the velocity direction (i.e. $\mathbf{v} = -\mathbf{v}$). The system would then start accelerating towards $\mathcal{S}$, and as soon as it gets inside $\mathcal{S}$ it could try to stop before leaving $\mathcal{S}$ again. The only problem with this approach is that, in theory, the algorithm could loop forever, e.g. if the system behaves like a damping-less pendulum. Properly dealing with these unusual cases would significantly increase the complexity of our algorithm. Since we prefer to keep the algorithm simple to make it more accessible to the community, we decided not to deal with this unusual case in this paper.

The decision tree of the algorithm is depicted in Fig. 3,

Algorithm 1 capturability algorithm.

function IS_STATE_CAPTURABLE($\mathbf{c}, \dot{\alpha}, \mathbf{v}, \mathbf{A}, m, \mathbf{g}$)
  $\alpha \leftarrow 0$
  $\dot{\alpha}_0 \leftarrow \dot{\alpha}$
  last_iteration $\leftarrow$ False
  for $i = 0$ to MAX_ITER **do**
    if $\dot{\alpha} = 0$ **then** ▷ Check for C3
      $\alpha_s \leftarrow$ compute_closest_static_alpha($\mathbf{c}, \alpha, \mathbf{v}, \mathbf{A}, \mathbf{g}$)
      if $\alpha = \alpha_s$ **then** ▷ Check for C3.1
        return True
      if $\alpha < \alpha_s$ **then** ▷ Check for C3.2
        return is_static_region_reachable($\mathbf{c}, \dot{\alpha}_0, \mathbf{v}, \mathbf{A}, m, \mathbf{g}$)
      return CaseNotHandled ▷ C3.3
    if last_iteration **then**
      return False
    (feasible, $\beta, \gamma, \bar{\alpha}$) $\leftarrow$ compute_min_acc($\mathbf{c}, \mathbf{v}, \mathbf{A}, m, \mathbf{g}$)
    if not feasible **then** ▷ Check for C2
      return False
    if $\gamma > 0$ and $\beta + \gamma\bar{\alpha} > 0$ **then** ▷ Check for C1
      $\bar{\alpha} \leftarrow -\frac{\beta}{\gamma}$
      last_iteration $\leftarrow$ True
    $(\alpha, \dot{\alpha}) \leftarrow$ integrate_LDS($\alpha, \dot{\alpha}, \beta, \gamma, \bar{\alpha}$)

Algorithm 2 Algorithm to check whether static region is reachable.

function IS_STATIC_REGION_REACHABLE($\mathbf{c}, \dot{\alpha}, \mathbf{v}, \mathbf{A}, m, \mathbf{g}$)
  $\alpha \leftarrow 0$
  last_iteration $\leftarrow$ False
  for $i = 0$ to MAX_ITER **do**
    if $\dot{\alpha} = 0$ **then** ▷ Check for C3.2.2
      return False
    if last_iteration **then**
      return True
    (feasible, $\beta, \gamma, \bar{\alpha}$) $\leftarrow$ compute_max_acc($\mathbf{c}, \mathbf{v}, \mathbf{A}, m, \mathbf{g}$)
    if $\beta + \gamma\bar{\alpha} \geq 0$ **then** ▷ Check for C3.2.1
      last_iteration $\leftarrow$ True
    $(\alpha, \dot{\alpha}) \leftarrow$ integrate_LDS($\alpha, \dot{\alpha}, \beta, \gamma, \bar{\alpha}$)

while the whole algorithm is summarized by Algorithms 1 and 2. In the following we present a proof of convergence of the algorithm (Section III-C) and we discuss a simple extension to quantify how close the given state is to falling (Section III-D).

The function *compute_closest_static_alpha* (used in Algorithm 1) is described in Appendix A. In a few words, it solves a Quadratic Program to determine where the given CoM position is with respect to the static-equilibrium region. The function *integrate_LDS* is described in Appendix B. Finally, the function *compute_min_acc* is described in Appendix C.

C. Proof of convergence

Theorem. *Algorithm 1 terminates in a finite number of iterations by one of the cases of Fig. 3.*

Proof. Consider an arbitrary initial state $(\alpha, \dot{\alpha})$, corresponding to a minimal acceleration (maximal deceleration) $\ddot{\alpha}_{min}$. If no corresponding acceleration exists, then the algorithm immediately terminates with scenario C2. Otherwise, at each iteration the algorithm follows an edge of the convex polygon $\alpha, \ddot{\alpha}$ (which may be open) until either another edge is found, or one of the conditions of Fig. 3 is met. The current edge might be bounded by another edge $\bar{\alpha}$, by the axis $\ddot{\alpha} = 0$, or it might be unbounded (below). In the first case, a new edge is reached, which corresponds either to a new iteration or to scenario C2 (termination with negative answer). The second case corresponds to scenario C1 (termination with negative answer). In the last case, the system can always decelerate, thus it can reach $\dot{\alpha} = 0$ (scenario C3). Since the polygon has a finite number of edges (upper bounded by the number of faces of the linearized centroidal wrench cone (6)), the main loop of Alg. 1 is guarantee to terminate in one of the scenarios of Fig. 3.

To conclude the proof we have to check that the second loop (scenario C3.2) integrating $\ddot{\alpha}_{max}$ converges as well. With similar arguments, the current edge is either bounded by a new edge, by the axis $\ddot{\alpha} = 0$ or unbounded. This corresponds either to a new iteration, to termination with C3.2.1 or to the guarantee to be able to reach C3.2.2. □

D. Approximate Capturability Margin

Rather than merely predicting whether the system is going to fall, we could measure how close it is to falling. This information can be useful for controller design or to evaluate the risk of fall. In case the algorithm terminates with a negative answer, the final CoM velocity (i.e. $\dot{\alpha}$) can be used as an approximate distance of the current state to the capturability kernel. If instead the algorithm terminates with a positive answer, we could measure how much additional initial CoM velocity the system could have handled. However, this measure would require additional computations, hence a longer computation time. We propose instead to use another measure, which is correlated to this one, but that comes at zero computational cost. Once the system reaches the final state $\dot{\alpha}_{final} = 0$, we take the maximum deceleration $\ddot{\alpha}_{min}(\alpha_{final})$ as an approximate distance of the current state to the borders of the capturability kernel. In the case of coplanar contacts this value is actually proportional to the distance of the capture point to the support polygon borders, so it seems a reasonable way to approximate the capturability margin.

IV. Capture Point Extension

In order to evaluate our capturability algorithm we would like to compare it against other capturability algorithms for multi-contact scenarios. However, classic capturability tools (such as the capture point) are not designed to work in multi-contact. In this section we present a simple extension of the capture point, which drastically improves its performance in multi-contact.

The (instantaneous) capture point is the 2d point on the ground where the robot has to step to come to a stop [1], [2]. Using the dynamics of the LIPM we can easily derive the analytical expression of the capture point:

$$\mathbf{c}^{xy} + \frac{\dot{\mathbf{c}}^{xy}}{\omega}, \tag{14}$$

where $\omega = \sqrt{g/c^z}$, g is the gravity acceleration, and c^z is the height of the CoM. The capture point has been originally introduced for push recovery [3]. However, later it has been also used as a criterion for zero-step capturability [26]. This is based on the simple observation that, as long as the capture point remains inside the convex hull of the contact point, the robot state is capturable.

Our tests will empirically demonstrate that this approach no longer works in multi-contact scenarios. However, we can modify this criterion to account for the additional contacts. We suggest to use the support polygon [22] rather than the convex hull of the contact points. It is well-known that, in case of coplanar contacts, the two polygons are equivalent. This is no longer the case in multi-contact scenarios.

Even though this approach is heuristic, and mainly based on our intuition, our tests show that it gives rather good results in practice—although not as good as our algorithm.

V. Results

This section presents simulation results with the 36-degree-of-freedom humanoid robot HRP-2 [27] in a push-recovery scenario. The goals of our tests were:

1) to measure how accurately our algorithm can predict the fall of a complex legged robot (Section V-C);
2) to compare our algorithm with other capturability margins (Section V-C);
3) to identify the major reasons of false prediction (Section V-D);
4) to compare the CoM paths found by our algorithm with the ones generated by the robot (Section V-E);
5) to assess whether our algorithm is sufficiently fast for online applications (Section V-F).

We initialized all simulations with random joint positions and velocities (Section V-B), and used our algorithm to predict whether a fall was inevitable. We then verified whether the prediction was correct by simulating the system using a balance controller (Section V-A). We repeated this process thousands of times, with different numbers of contacts: two (the feet), three (feet and one hand) and four (both feet and hands). Finally, we compared the accuracy of fall prediction of three different capturability margins:

- the one presented in this paper (Section III);
- the capture point distance to the support polygon, positive if it is inside, negative otherwise (Section IV);
- the capture point distance to the convex hull of the contact points projected on a plane orthogonal to gravity.

A. Balance Controller

Ideally the balance controller used for comparison should always be capable of avoiding a fall whenever this is possible (ground truth). However, we are not aware of any such controller. The closest approach to an ideal controller would probably be a whole-body trajectory optimization [28]. However, its large computation time prevents both extensive testing for validation in simulation and application on real systems for balance recovery. A common alternative is to optimize only a subpart of the robot dynamics, such as the centroidal dynamics [29]. Both whole-body and centroidal trajectory optimization boil down to nonconvex optimization problems, which thus extensively rely on either a good initial guess or a convex approximation. We are not aware of any of these that has been proven efficient in the difficult case of balance recovery.

Nowadays, standard balance controllers used on real systems are *local* controllers that try to stop the robot while satisfying all its dynamic constraints [30]. From a pragmatic point of view, it is interesting to evaluate how well our algorithm can predict the failure of these controllers—rather than of an ideal controller. For these reasons, we used a standard Task-Space Inverse Dynamics controller [26], which is described in details in Appendix D. To improve the quality of our ground truth, we actually considered the best between three different balance controllers. If the first controller failed to stop the robot, we tried the second one. If also the second one failed, we tried the third one. The controllers are simple variations of the same inverse-dynamics controller. The first controller is the standard one, described in Appendix D. The second one tries to maintain the CoM on a straight line. The third one uses a larger value of the maximum joint deceleration used for converting the joint position bounds into joint acceleration bounds (see Appendix D for more details).

B. Methodology

We decided to test our algorithm in a push-recovery scenario: the robot starts in an equilibrium configuration, and then an impulsive force instantaneously changes its joint velocities. At that point the balance controller tries to stop the CoM while maintaining the initial joint posture.

In order to get *reasonable* initial conditions we had to bias the random sampling in different ways. We first sampled the robot configuration $\mathbf{q}$, which had to satisfy the following constraints:

- The robot CoM is above the support polygon [22] (this is a necessary condition for static equilibrium)
- No self collision
- The feet are in contact with the ground (only for the test with two coplanar contacts)

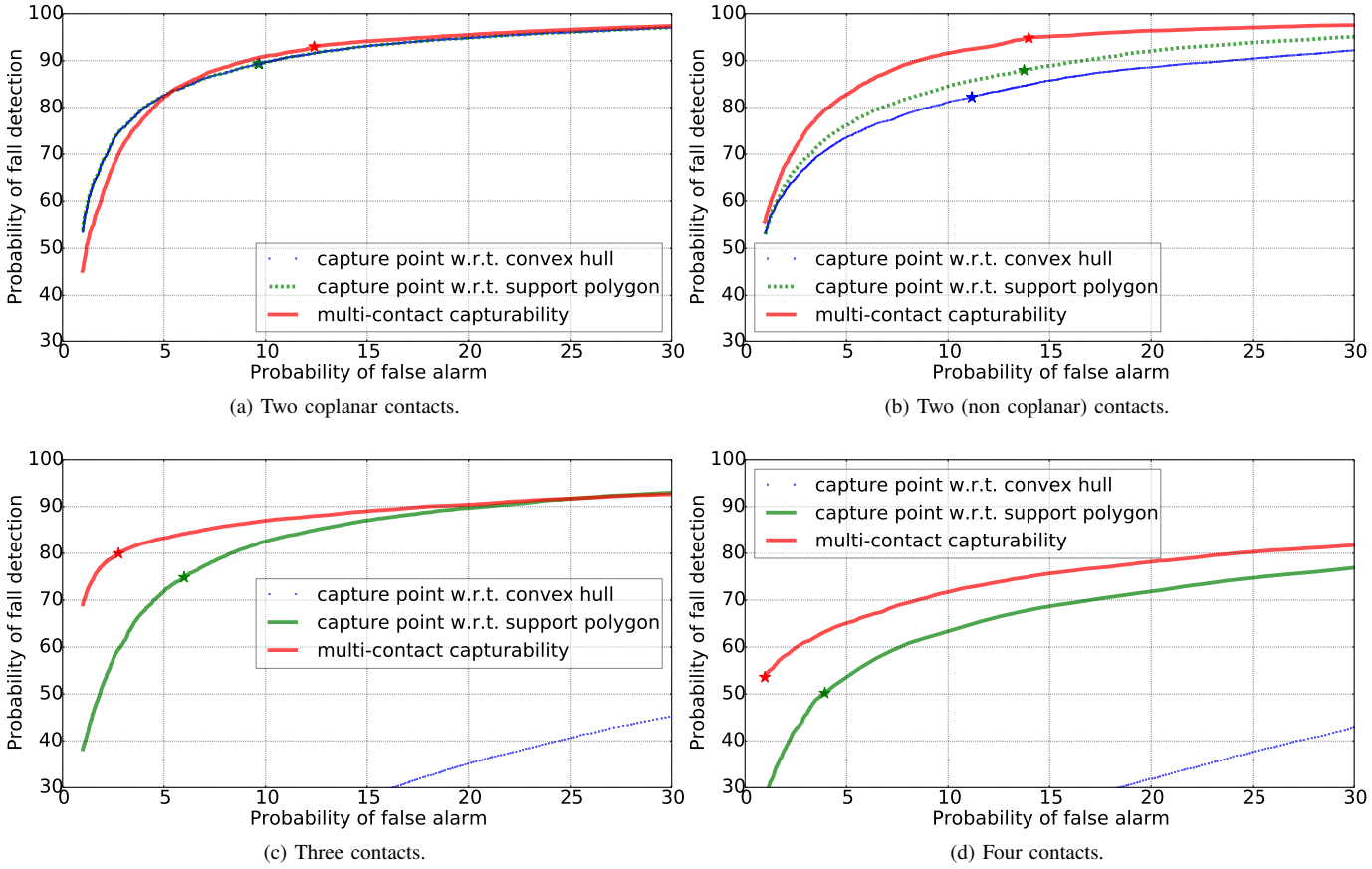
We then sampled the initial robot velocity vector $\mathbf{v}$, which had to satisfy the following linear constraints:

- Zero velocity at the contact points
- Each joint should be able to stop before hitting its position bounds by using a limited user-defined acceleration $\ddot{\mathbf{q}}_j^{max}$

To enforce the last constraint, each joint velocity must satisfy [31]:

$$-\sqrt{2\ddot{q}^{max}(q - q^{min})} \leq \dot{q} \leq \sqrt{2\ddot{q}^{max}(q^{max} - q)} \qquad (15)$$

The value of $\ddot{q}^{max}$ has been heuristically tuned to 10 rad/s^2. We carried out $4{\times}10^4$ tests for each number of contacts. Appendix E provides both theoretical and empirical evidence to justify this sample size. For the comparison we used the receiver operating characteristic (ROC) curve, a standard way to show the ability of a binary classifier as its discrimination threshold varies. The ROC curve shows the probability of



(a) Two coplanar contacts.

(b) Two (non coplanar) contacts.

(c) Three contacts.

(d) Four contacts.

Fig. 4. Receiver operating characteristic (ROC) curves. The stars represent the performance corresponding to a margin of zero.

detection (also known as "true positive rate" or "sensitivity") against the probability of false alarm (also known as "false positive rate" or "fall-out") at various threshold settings. In our context the probability of detection is computed as the number of times a fall has been correctly predicted over the number of times the robot fell. Similarly, the probability of false alarm is the number of times a fall has been erroneously predicted over the number of times the robot did not fall.

We considered the robot to be fallen if:

- the Quadratic Program of the controller became unfeasible, or
- the position of an end-effector in contact moved more than 10 cm from its initial position.

Otherwise, we considered the robot to have successfully avoided the fall as soon as $||\mathbf{v}|| < 0.01$.

C. Discussion

The results for 2 coplanar, 2 non coplanar, 3 and 4 contacts are summarized by the ROC curves in Fig. 4a, 4b, 4c and 4d, respectively.

As expected, for coplanar contacts the three capturability margins performed well and they are approximately equivalent—although our margin was slightly better, probably due to the fact that it accounts also for vertical CoM velocities. For two (non coplanar) contacts our capturability margin performed significantly better than the other two. Moreover, the capture point margin performed better when using the support polygon than when using the convex hull of the contact points. This is reasonable because the convex hull of the contact points is only a rough approximation of the support polygon when the contact points are not coplanar. It is interesting to observe that our algorithm performed the same for two coplanar and non-coplanar contacts. For both 3 and 4 contacts the capture point margin w.r.t the convex hull performed very poorly, while our capturability margin still outperformed the capture point margin w.r.t. the support polygon.

It is somehow surprising that the capture point margin w.r.t. the support polygon worked quite well even for 3 and 4 contacts. The capture point is based on the linear inverted pendulum model, which is known to break down in multi-contact situations. The reason why it worked in this context is that we used it in combination with the real support polygon, computed using techniques valid for multi-contact scenarios. This provides then an interesting alternative to our capturability margin, which is easier to code and faster to compute—even though it does not perform as good.

Finally, our capturability margin performed better for 2 contacts than for 3, and better for 3 contacts than for 4. This is reasonable because the higher the number of contacts, the more constrained the robot motion is, in a way that is not accounted for by our algorithm. However, we could try to account for these kinematic constraints in future work using

existing methods [29], [32].

D. Analysis of False Predictions

The aim of this subsection is to determine the major reason of false prediction of our algorithm. Before presenting our results, we discuss the potential issues of such an analysis.

1) Disclaimer: The difference between the real capturability problem and the approximated one (12) solved by our algorithm lies in the considered constraints. In the real problem we have the contact force friction cones, the contact constraints (i.e. contact points should not move), and the joint position, velocity and torque limits. Our algorithm instead is based on the centroidal dynamics, which only captures the friction cone constraints. Moreover, we imposed two constraints to simplify our problem: i) constant angular momentum, and ii) straight CoM path. Determining the reason of a false prediction boils down to determining which constraint (added or removed) is responsible for the false predictions. This is equivalent to determining which constraint is the cause of infeasibility of an optimization problem. There are two main issues when trying to do this.

First, we are dealing with a nonconvex problem, which we do not know how to solve in general, neither do we know how to prove its infeasibility. More importantly, the descent directions chosen by the solver depend on the problem constraints. For this reason, it can happen that by removing some constraints the solver is no longer able to find a solution for a problem that it was able to solve before. Since in theory a feasible problem cannot become unfeasible by removing a constraint, this fact is hard to take into account in an analysis.

Second, in general, an optimization problem is not unfeasible due to a constraint, but to a combination of constraints. For instance, a problem may be feasible if considering either the torque limits or the position limits, but unfeasible if considering both. In these cases, it is hard to say what the "reason" of false prediction is.

Because of these two issues, any analysis that tries to identify the reasons of false predictions could lead to inconsistent conclusions, thus its results should be viewed with skepticism. In spite of all these potential problems, we present here the results of our analysis.

2) Approach: Our approach consisted in adding/removing constraints to/from the robot dynamics, to make it "closer" to the simplified dynamics used by our algorithm. We then repeated the simulations with the balance controller and computed the number of false predictions. We then assume that the major cause of false predictions is the constraint that, when removed, leads to the largest improvement. In our first test we removed the following (whole-body) constraints:

- joint torque limits (No tau lim)
- joint position limits (No q lim)
- joint torque and position limits (No q-tau lim)
- joint position and velocity limits (No q-dq lim)
- joint position, velocity and torque limits (No lim)

In our second test instead we added the following (centroidal) constraints:

- constant angular momentum (No ang mom)

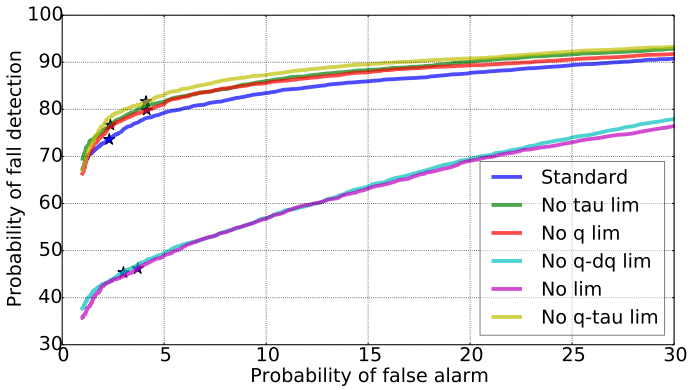


Fig. 5. ROC curves of our algorithm, using different controllers to compute the ground truth. In each controller we removed some combination of the whole-body constraints that are neglected by our algorithm, i.e. the joint torque, position, and velocity limits. We computed these curves using 10^4 samples with 3 contacts.

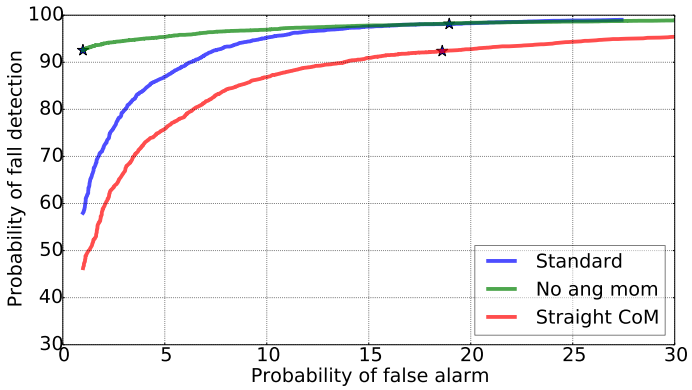
- CoM acceleration parallel to velocity (Straight CoM)

In order to force a constant angular momentum throughout the simulation we had to enforce zero angular momentum at the beginning, to allow the robot to come to a stop.

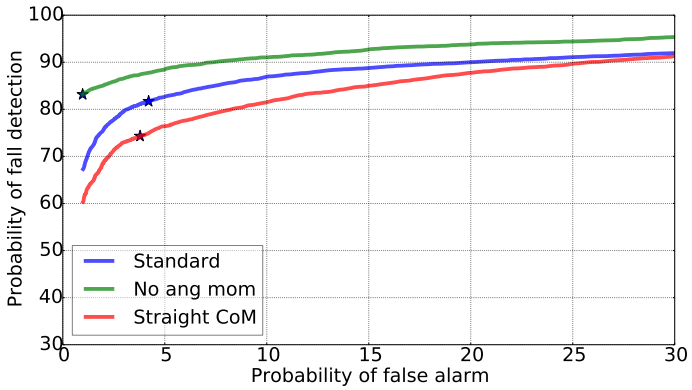
This analysis was expensive in terms of computation time and storage because it requires to repeat each test with eight controllers (i.e. the standard one plus the seven controllers listed above), so we performed it only for the cases of two or three contacts, and a smaller sample size (i.e. 10^4).

3) Whole-Body Constraints: Fig. 5 shows the ROC curves obtained by removing the above-mentioned whole-body constraints. Removing the joint torque or position limits had a positive effect on the prediction accuracy of our algorithm. For a margin of zero (i.e. the stars in Fig. 5), the increase of the fall detection probability was higher for the position limits (from 74% to 80%) than for the torque limits (from 74% to 77%). However, the increase of the false alarm probability was also higher for the position limits (from 2.3% to 4%) than for the torque limits (from 2.3% to 2.4%). For this reason, the two ROC curves for no torque and no position limits are almost equivalent. Removing both torque and position limits resulted in a slightly better prediction than removing only one of the two (detection probability 81%, false alarm probability 4%). On the contrary, removing the joint position and velocity limits led to an unexpected behavior of the controller. The controller did not manage to prevent the fall in about half of the tests in which the standard controller succeeded. The reason for this behavior is that the absence of both position and velocity limits allows the controller to arbitrarily exploit the angular momentum to decelerate the CoM faster. This leads the robot into states with extremely high velocities, in which it becomes impossible to satisfy the contact constraints. This behavior prevented us from determining whether the joint velocity limits are a major cause of false prediction.

4) Centroidal Constraints: Fig. 6 shows instead the ROC curves obtained by adding the above-mentioned centroidal constraints. Our algorithm was able to predict much better the outcome of the controller that was forced to maintain a null angular momentum. For two contacts, the probability of

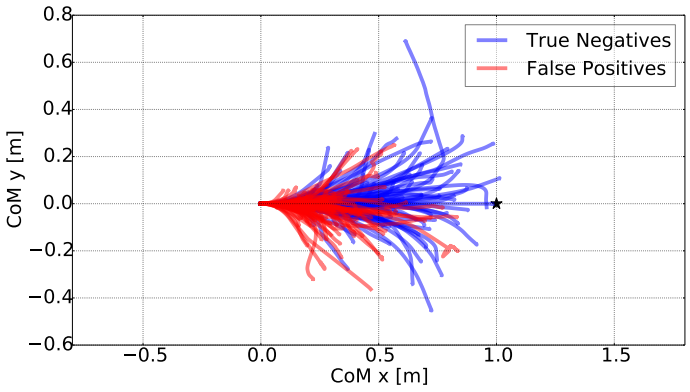


(a) Two contacts.

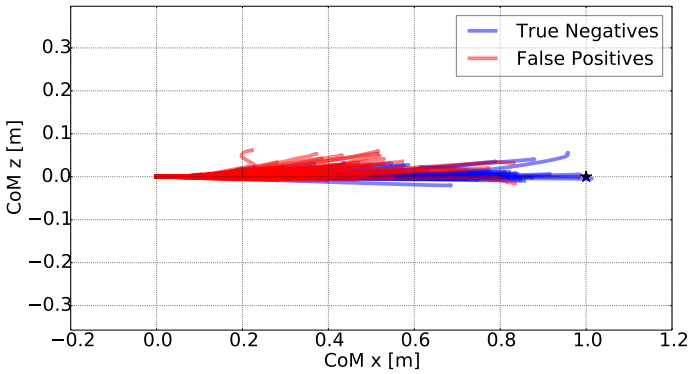


(b) Three contacts.

Fig. 6. ROC curves of our algorithm, using different controllers to compute the ground truth. In each controller we imposed one of the two constraints used by our algorithm to simplify the capturability problem: i) constant angular momentum, or ii) straight-path CoM. We computed these curves using 10^4 samples.



(a) xy plane.



(b) xz plane.

Fig. 7. Randomly selected CoM paths generated by the balance controller with 2 contacts: 100 samples for capturable states correctly predicted by our algorithm (i.e. True Negatives), and 100 samples for capturable states incorrectly predicted by our algorithm (i.e. False Positives).

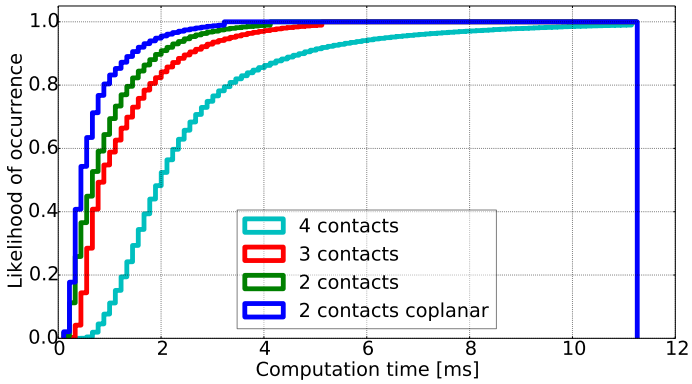
false alarm decreased much more (from 19% to 1%) than the probability of fall detection (from 98% to 92%). This suggests that we can expect significant improvements by accounting for angular momentum in our algorithm (e.g. modeling a flywheel with bounded torque and angle). On the contrary, forcing the CoM path to be straight affected much more the probability of fall detection (from 98% to 92%) than the probability of false alarm (from 19% to 18.6%). The small change of the probability of false alarm suggests that the "straight CoM path" hypothesis is not the main reason of the false alarms of our algorithm. The significant decrease of the probability of fall detection instead is due to conflicts between the whole-body constraints and the "straight CoM path" constraint. For three contacts the results are similar. The main difference is that the probability of false alarm of the standard controller is much lower than for two contacts (4% instead of 19%), thus the improvement led by the "constant angular momentum" constraint is smaller. This is likely due to the reduced capabilities to generate angular momentum when a third contact is added.

E. CoM Paths

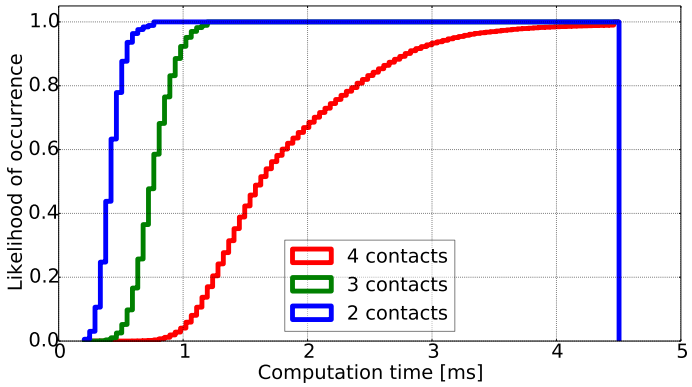
In this subsection we compare the CoM trajectories generated by the controller with the ones predicted by our algorithm. Fig. 7 shows 200 randomly selected CoM paths generated by our balance controller. Half of these paths correspond to True Negative cases, the other half correspond to False Positive cases. In all these tests the robot was in contact with both feet. We scaled and rotated these paths so that the path predicted by our algorithm always starts at (0,0,0) and ends at (1,0,0), which corresponds to the star sign in Fig. 7. In general, the controller was able to stop the CoM traveling less distance than what predicted—even as little as 20% of the predicted distance. This is reasonable considering that our prediction is based on the conservative assumption of constant angular momentum. Interestingly, it seems that on average the CoM paths are longer for the True Negative cases than for the False Positive cases. We can clearly see that in general the controller did not generate straight CoM paths, but these paths do not differ too much from straight lines, especially in the xz plane. Note that the z direction in Fig. 7 is not the gravity direction, but it is the direction orthogonal to the initial CoM velocity and the y axis of the world frame.

F. Computation Time

We implemented our algorithm in python, but we used a C++ solver for the LPs (to compute the minimum CoM accelerations). Since the LPs are the most computationally expensive part of our algorithm, we report here only the time



(a) Multi-contact capturability algorithm.



(b) Capture point distance to support polygon. We do not show the coplanar case because it could be implemented using simpler algorithms (i.e. 2d convex hull) that are orders of magnitude faster.

Fig. 8. Cumulative histograms of the computation times of the capturability algorithms, measured on a 64-bit Ubuntu machine, with a 3.4 GHz CPU and 16 GB of RAM. These histograms show the likelihood that the computation time be less than a certain value.

taken to solve the LPs. With a complete C++ implementation the total time would be only marginally higher.

Fig. 8 shows the computation times of our algorithm and the capture-point algorithm (using the support polygon[1]) for different numbers of contacts. As expected, the time increased with the number of contacts. This is due to the increased size of the LPs. Remarkably, it never exceeded 11 ms, and most of the times it was below 5 ms. This efficiency is crucial for a fall prediction scenario because it allows for fast reactions. The capture-point algorithm is faster than ours because it only needs to solve one LP to compute the distance between the capture point and the support polygon borders. Our algorithm instead may have to solve several LPs of similar complexity to find the minimum CoM accelerations to integrate.

VI. Conclusions and Future Work

This paper presented a fast algorithm to compute an approximate capturability margin for legged robots in multi-contact scenarios. The algorithm relies on the point-mass model and the simplifying assumption that, in order to come to a stop, the robot only needs to accelerate its CoM in the direction opposite to its initial velocity. This assumption is somewhat conservative, but it makes our problem convex, allowing us to solve it efficiently (i.e. in less than 11 ms).

We validated our approach performing thousands of simulations with the HRP-2 humanoid robot, in different random multi-contact scenarios (using 2, 3, or 4 contacts). We implemented a state-of-the-art inverse-dynamics balancing controller, and we evaluated the ability of our algorithm to predict the outcome of the simulations. The performance varied depending on the number of contacts (the lower, the better). For instance, with 2 contacts our algorithm was able to predict a fall with a probability of 90%, while having a probability of false alarm of about 8%. Regardless the number of contacts, our algorithm outperformed the capture point margin w.r.t. the convex hull of the contact points, which performed very poorly in the multi-contact case. Our results also show that the capture point margin w.r.t. the support polygon is a reasonable fall prediction indicator, even in multi-contact scenarios, although less accurate than our criterion.

An interesting direction for future work would be the inclusion of the presented capturability criterion inside an inverse dynamics controller—similarly to [26], but for multi-contact scenarios. This would allow for the generation of arbitrary movements, while guaranteeing the balance of the robot. However, this extension does not seem trivial. The presented algorithm determines the membership of a given state to the capturability kernel, but without explicitly computing the kernel. This begs the question of how to include the capturability constraint inside the controller. While the capturability kernel could be approximated offline through a sampling-based approach, we expect it not to be a convex set. It is then unclear how this nonconvex constraint should be included in the convex QP solved by state-of-the-art inverse dynamics controllers.

This paper discussed the problem of zero-step capturability. An obvious extension would be to deal with the more general problem of N-step capturability. However, the assumption that the CoM moves on a straight line would be too conservative for $N>0$. When the CoM projection leaves the support polygon we can no longer accelerate it in all directions, which may prevent us from maintaining it on a straight line. A possible alternative could be to assume that the CoM remains on a plane, which would confine the nonlinearity of the centroidal wrench to its last element only. This nonlinearity could then be treated using robust optimization techniques, in the same spirit as [33], or by simply neglecting it [34].

Our analysis of Section V-D suggests that neglecting angular momentum is the major reason of false alarms of our algorithm. We could thus expect large improvements by including a flywheel in our model, which would allow for a bounded generation of angular momentum. However, it is not clear how to connect the orientation of this flywheel with the orientation of the different bodies of the robot, given the nonintegrability of the average angular velocity [35].

[1]We do not report the computation times for the other capture-point algorithm because it is known that convex hull techniques for the 2d case are orders of magnitude faster.

APPENDIX A
COMPUTE CLOSEST STATIC ALPHA

This section describes the function *compute_closest_static_alpha* (used in Algorithm 1). We need to compute a value α_s as close as possible to a given reference α, such that the CoM position $\mathbf{c}+\alpha_s\mathbf{v}$ allows for static equilibrium. This can be achieved by solving the following Quadratic Program:

$$\begin{aligned} \underset{\alpha_s,\mathbf{f}}{\text{minimize}} \quad & ||\alpha_s - \alpha||^2 \\ \text{subject to} \quad & \begin{bmatrix} -m\mathbf{g} \\ m\mathbf{g}\times\mathbf{c}_0 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ m\mathbf{g}\times\mathbf{v} \end{bmatrix}\alpha_s = \mathbf{A}\mathbf{f} \\ & \mathbf{B}\mathbf{f} \leq \mathbf{0} \end{aligned} \tag{16}$$

If the resulting value of α_s is equal to α, then the CoM position $\mathbf{c}+\alpha\mathbf{v}$ belongs to the support polygon. If instead $\alpha_s < \alpha$, then the CoM is located after the support polygon (according to the direction $\mathbf{v}$). Otherwise $\alpha_s > \alpha$, which means that the CoM is located before the support polygon. This QP is unfeasible only if the line $\mathbf{c}+\alpha_s\mathbf{v}$ does not intersect the support polygon. In this case, the initial state $(\mathbf{c}, \dot{\mathbf{c}})$ is labeled as not capturable.

APPENDIX B
INTEGRATION OF PIECEWISE-LINEAR DYNAMICAL SYSTEM (PLDS)

This appendix explains in details how to integrate the PLDS. Given the interval of linearity $[\underline{\alpha}, \bar{\alpha}]$ (defined in (13), but used here without index i to improve readability) that contains the current value of α, and the values β, γ defining $\ddot{\alpha}$ as an affine function of α, we have to integrate the following LDS:

$$\frac{d}{dt}\begin{bmatrix}\alpha(t)\\ \dot{\alpha}(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ \gamma & 0\end{bmatrix}\begin{bmatrix}\alpha(t)\\ \dot{\alpha}(t)\end{bmatrix} + \begin{bmatrix}0\\ \beta\end{bmatrix} \tag{17}$$

We want to integrate until one of these two conditions is met: i) $\alpha(t) = \bar{\alpha}$, or ii) $\dot{\alpha}(t) = 0$. The termination conditions on $\ddot{\alpha}$ mentioned in Section III-B can be handled by properly modifying $\bar{\alpha}$ before starting the integration (see line 19 of Algorithm 1).

The explicit solution of this LDS can take two different forms, depending on whether γ is null. We deal with them separately in the following two subsections.

1) Acceleration depends on position: If $\gamma \neq 0$ (which is the typical case) then the explicit solution of this system is [36]:

$$\begin{bmatrix}\alpha(t)\\ \dot{\alpha}(t)\end{bmatrix} = \begin{bmatrix}\cosh(\omega t) & \frac{\sinh(\omega t)}{\omega}\\ \omega\sinh(\omega t) & \cosh(\omega t)\end{bmatrix}\begin{bmatrix}\alpha(0)\\ \dot{\alpha}(0)\end{bmatrix} + \begin{bmatrix}\cosh(\omega t) - 1\\ \omega\sinh(\omega t)\end{bmatrix}\frac{\beta}{\gamma}, \tag{18}$$

where $\omega = \sqrt{\gamma}$. When γ is negative, ω is an imaginary number, but $\alpha(t)$ and $\dot{\alpha}(t)$ always remain real numbers. Since we have to integrate until $\alpha(t) = \bar{\alpha}$, or $\dot{\alpha}(t) = 0$, we need to know the time at which these events will occur. We can compute the time at which $\dot{\alpha}(t) = 0$ by using the second line of (18).

$$\begin{aligned} &\omega\sinh(\omega t)\alpha(0) + \cosh(\omega t)\,\dot{\alpha}(0) + \omega\sinh(\omega t)\frac{\beta}{\gamma} = 0 \\ &t = \frac{1}{\omega}\operatorname{atanh}\left(\frac{-\dot{\alpha}(0)}{\omega(\alpha(0) + \beta/\gamma)}\right) \triangleq t_{zv} \end{aligned} \tag{19}$$

If the argument of atanh does not belong to the interval $[-1, 1]$ it means that $\dot{\alpha}$ will never be zero. Otherwise, we have to verify that the position limit is not reached before t_{zv}, that is: $\alpha(t_{zv}) \leq \bar{\alpha}$. If that is the case, the algorithm terminates. Otherwise, this means that α will reach $\bar{\alpha}$ with a positive velocity.

We can compute the time at which $\alpha(t) = \bar{\alpha}$ by using the first line of (18):

$$\begin{aligned} &\cosh(\omega t)\alpha(0) + \frac{1}{\omega}\sinh(\omega t)\,\dot{\alpha}(0) + (\cosh(\omega t) - 1)\frac{\beta}{\gamma} = \bar{\alpha} \\ &t = \begin{cases} \frac{1}{\omega}\log\left(\frac{-C+\sqrt{B^2+C^2-A^2}}{A+B}\right), & \text{if } \gamma > 0 \\ \frac{1}{\omega}\log\left(\frac{-C-\sqrt{B^2+C^2-A^2}}{A+B}\right), & \text{if } \gamma < 0 \end{cases} \end{aligned} \tag{20}$$

where:

$$A = \alpha(0) + \frac{\beta}{\gamma}, \qquad B = \frac{\dot{\alpha}(0)}{\omega}, \qquad C = -\bar{\alpha} - \frac{\beta}{\gamma} \tag{21}$$

The logarithm in the expression of t is (in general) a complex logarithm. However, t is always a real number.

2) Acceleration does not depend on position: When $\gamma = 0$, the solution of our LDS is:

$$\begin{bmatrix}\alpha(t)\\ \dot{\alpha}(t)\end{bmatrix} = \begin{bmatrix}1 & t\\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha(0)\\ \dot{\alpha}(0)\end{bmatrix} + \begin{bmatrix}\frac{1}{2}t^2\\ t\end{bmatrix}\beta \tag{22}$$

In this case we can easily compute the time at which $\dot{\alpha}(t) = 0$ as:

$$t_{zv} = -\frac{\dot{\alpha}(0)}{\beta} \quad \rightarrow \quad \alpha(t_{zv}) = \alpha - \frac{\dot{\alpha}(0)^2}{2\beta} \tag{23}$$

As before, we need then to check whether $\alpha(t_{zv}) \leq \bar{\alpha}$. If this condition is satisfied the algorithm terminates. If that is not the case, we have to compute the time at which $\alpha(t) = \bar{\alpha}$. Since the position trajectory is a parabola, there exist two values of t such that $\alpha(t) = \bar{\alpha}$. We take the smallest of the two because we are interested in the first time where $\alpha(t)$ reaches $\bar{\alpha}$:

$$t = \frac{-\dot{\alpha}(0) + \sqrt{\dot{\alpha}(0)^2 - 2\beta(\alpha(0) - \bar{\alpha})}}{\beta} \tag{24}$$

Algorithm 3 summarizes the integration of the $(\alpha, \dot{\alpha})$ linear dynamical system with input acceleration $\ddot{\alpha} = \beta + \gamma\alpha$, until either $\dot{\alpha}(t) = 0$ or $\alpha(t) = \bar{\alpha}$.

APPENDIX C
COMPUTING ACCELERATION BOUNDS

We already saw in Section III-A that we can compute the $(\alpha, \ddot{\alpha})$ polytope by means of the centroidal cone matrix $\mathbf{H}$. However, computing $\mathbf{H}$ can be computationally expensive (about 5-10 ms), and being fast is critical in the context of predicting a fall. We thus propose an alternative method to compute $\ddot{\alpha}_{min}$, which resulted to be computationally faster in our tests. From (3), (5) and (11) we can easily see that, for a given value of α, we can compute $\ddot{\alpha}_{min}$ by solving the following Linear Program (LP):

$$\begin{aligned} \underset{\mathbf{x}=(\ddot{\alpha},\mathbf{f})}{\text{minimize}} \quad & \ddot{\alpha} \\ \text{subject to} \quad & \begin{bmatrix} m\mathbf{I} \\ m\,\mathbf{c}\times\mathbf{v} \end{bmatrix}\ddot{\alpha} + \begin{bmatrix} -m\mathbf{g} \\ m\mathbf{g}\times\mathbf{c}_0 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ m\mathbf{g}\times\mathbf{v} \end{bmatrix}\alpha = \mathbf{A}\mathbf{f} \\ & \mathbf{B}\mathbf{f} \leq \mathbf{0} \end{aligned} \tag{25}$$

Algorithm 3 Algorithm to integrate the Linear Dynamical System.

function INTEGRATE_LDS($\alpha, \dot{\alpha}, \beta, \gamma, \bar{\alpha}$)
  if $\gamma = 0$ **then**
    $\alpha_t \leftarrow \alpha - 0.5\,\dot{\alpha}^2\,/\beta$
    if $\alpha_t \leq \bar{\alpha}$ **then**
      return $(\alpha_t, 0)$
    $t \leftarrow \frac{-\dot{\alpha} + \sqrt{\dot{\alpha}^2 - 2\beta(\alpha - \bar{\alpha})}}{\beta}$
    return $(\bar{\alpha}, \dot{\alpha} + t\beta)$
  $\omega \leftarrow \sqrt{\gamma}$
  arg $\leftarrow \frac{-\dot{\alpha}}{\omega(\alpha + \beta/\gamma)}$
  if $|\text{arg}| \leq 1$ **then**
    $t \leftarrow \frac{1}{\omega}$ atanh(arg)
    $\alpha_t \leftarrow \cosh(\omega t)\alpha + \sinh(\omega t)\,\dot{\alpha}\,/\omega + (\cosh(\omega t) - 1)(\beta/\gamma)$
    if $\alpha_t \leq \bar{\alpha}$ **then**
      return $(\alpha_t, 0)$
  $(A, B, C) \leftarrow (\alpha_0 + \frac{\beta}{\gamma}, \frac{\dot{\alpha}_0}{\omega}, -\bar{\alpha} - \frac{\beta}{\gamma})$
  if $\gamma > 0$ **then**
    $t = \frac{1}{\omega}\log((\sqrt{B^2 + C^2 - A^2} - C)/(A + B))$
  else
    $t = \frac{1}{\omega}\log((-\sqrt{B^2 + C^2 - A^2} - C)/(A + B))$
  $\dot{\alpha}_t \leftarrow \omega\sinh(\omega t)\alpha + \cosh(\omega t)\,\dot{\alpha} + \omega\sinh(\omega t)(\beta/\gamma)$
  return $(\bar{\alpha}, \dot{\alpha}_t)$

Exploiting simple properties of LPs [37], the solution $\ddot{\alpha}_{min}$ can be written as a linear function of the problem parameter α:

$$\ddot{\alpha}_{min}(\alpha) = \beta + \gamma\alpha, \qquad \underline{\alpha} \leq \alpha \leq \bar{\alpha}, \tag{26}$$

where $\beta, \gamma, \underline{\alpha}$ and $\bar{\alpha}$ can be deduced from the problem solution. Once the LP optimum $\mathbf{x}^*$ has been computed, we know which constraints are active at the optimum by looking at the Lagrange multipliers. We then can collect all the active constraints in the matrix $\mathbf{K}$ and the vectors $\mathbf{k}_1, \mathbf{k}_2$ (which will include all the equalities and some of the inequalities) such that:

$$\mathbf{K}\mathbf{x}^* = \mathbf{k}_1\alpha + \mathbf{k}_2 \tag{27}$$

Since $\mathbf{K}$ is always a square invertible matrix, we can compute $\mathbf{x}^*$ as a function of α:

$$\mathbf{x}^*(\alpha) = \mathbf{K}^{-1}(\mathbf{k}_1\alpha + \mathbf{k}_2) \tag{28}$$

This expression remains valid as long as the active constraints do not change. We can verify this by using this expression:

$$\mathbf{B}\mathbf{S}_f\mathbf{K}^{-1}(\mathbf{k}_1\alpha + \mathbf{k}_2) \leq 0, \tag{29}$$

where $\mathbf{S}_f$ is a selection matrix such that $\mathbf{f} = \mathbf{S}_f\mathbf{x}$. By normalizing the rows of (29) we can easily find the upper and lower bounds of α, namely $\underline{\alpha}$ and $\bar{\alpha}$. Moreover, the first element of the vector $\mathbf{K}^{-1}\mathbf{k}_1$ is the derivative of $\ddot{\alpha}_{min}$ with respect to α (which we previously called γ). The same procedure can be applied for computing $\ddot{\alpha}_{max}$.

APPENDIX D
BALANCE CONTROLLER

The balance controller used in our tests is a standard Task-Space Inverse Dynamics controller [26], formulated as a Quadratic Program:

$$\begin{aligned}
\underset{\mathbf{x}=(\dot{\mathbf{v}},\mathbf{f},\boldsymbol{\tau})}{\text{minimize}} \quad & ||\mathbf{D}\mathbf{x} - \mathbf{d}||^2 \\
\text{subject to} \quad & \begin{bmatrix} \mathbf{J}(\mathbf{q}) & \mathbf{0} & \mathbf{0} \\ \mathbf{M}(\mathbf{q}) & -\mathbf{J}(\mathbf{q})^\top & -\mathbf{S}^\top \end{bmatrix} \begin{bmatrix} \dot{\mathbf{v}} \\ \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} -\dot{\mathbf{J}}(\mathbf{q},\mathbf{v})\mathbf{v} \\ -\mathbf{h}(\mathbf{q},\mathbf{v}) \end{bmatrix} \\
& \mathbf{B}\mathbf{f} \leq \mathbf{0} \\
& -\boldsymbol{\tau}_{max} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max} \\
& \dot{\mathbf{v}}_{min} \leq \dot{\mathbf{v}} \leq \dot{\mathbf{v}}_{max},
\end{aligned} \tag{30}$$

where n is the number of joints, $\mathbf{q} = (\mathbf{x}_b, \mathbf{q}_j) \in SE(3) \times \mathbb{R}^n$ are the robot base and joint configurations, $\mathbf{v} = (\mathbf{v}_b, \dot{\mathbf{q}}_j) \in \mathbb{R}^{n+6}$ are the base and joint velocities, $\boldsymbol{\tau} \in \mathbb{R}^n$ are the joint torques, $\mathbf{J} \in \mathbb{R}^{k\times(n+6)}$ is the constraint Jacobian, $\mathbf{M} \in \mathbb{R}^{(n+6)\times(n+6)}$ is the mass matrix, $\mathbf{h} \in \mathbb{R}^{n+6}$ contains the bias forces and $\mathbf{S} \in \mathbb{R}^{n\times(n+6)}$ is the selection matrix. The constraints $\mathbf{B}\mathbf{f} \leq \mathbf{0}$ represent a linearization of the friction cone constraints. Note that to maximize the accuracy of our fall prediction, the same cone linearization should be used in the controller and in the prediction algorithm.

We have used realistic joint velocity and torque limits in our simulations, as provided by the robot manufacturer. The velocity limits range from 2.2 to 9.1 rad/s. The torque limits range from 50 to 280 Nm (not considering wrist, gripper and neck joints, which have much lower torque limits). The joint-acceleration bounds have been computed so as to avoid violating the bounds of the joint positions and velocities [38]. We have tried different values of the maximum joint acceleration used to transform the joint position bounds into joint acceleration bounds: either 10 or 30 rad/s^2.

The cost function (defined by $\mathbf{D}$ and $\mathbf{d}$) represents the error of the tasks, that is the 2-norm of the difference between desired and actual task-space accelerations. We formulated the task of balancing by using two sub-tasks:

- stop the CoM: $\ddot{\mathbf{c}}^{des} = -k_d^{com}\,\dot{\mathbf{c}}$
- maintain the initial joint posture $\mathbf{q}_j^0$: $\ddot{\mathbf{q}}_j^{des} = k_p^j(\mathbf{q}_j^0 - \mathbf{q}_j) - k_d^j\dot{\mathbf{q}}_j$

We set $k_d^{com} = dt^{-1}$, so as to ask for zero CoM velocity in a single time step dt (which was set to 1 ms in our tests). The gains of the postural task instead were $k_p^j = 30$, and $k_d^j = 2\sqrt{k_p^j}$. As typically done, we gave higher priority to the CoM task by weighting its error 10^3 times more than the postural task error. We also tried adding a task to regulate the angular momentum to zero (as suggested in [30]), but we found that it was overall detrimental to the balancing performance, so we did not use it in the end.

APPENDIX E
SAMPLE SIZE

This appendix provides theoretical and empirical evidence to justify the sample size used for our statistical analysis. Our binary classification (capturable, or non-capturable) is based on the continuous *capturability margin* x. Given a threshold parameter x_{thr}, each test is classified as "capturable" if $x > x_{thr}$, and "non-capturable" otherwise. The variable x follows a probability density $f_1(x)$ if x actually belongs to

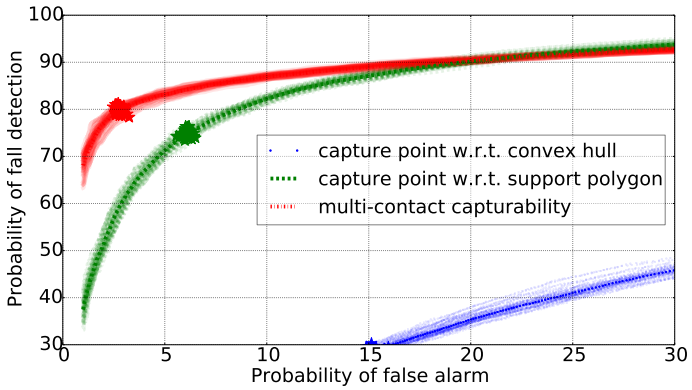


Fig. 9. Multiple ROC curves for 3 contacts. Each of the 50 curves is computed using 10^4 samples, randomly selected from a batch of 10^5 tests.

class "capturable", and $f_0(x)$ otherwise. Therefore, the True Positive Rate (TPR) and the False Positive Rate (FPR) are given by:

$$\text{TPR}(x_{thr}) = \int_{x_{thr}}^{\infty} f_1(x)\,dx \qquad \text{FPR}(x_{thr}) = \int_{x_{thr}}^{\infty} f_0(x)\,dx \tag{31}$$

The ROC curve plots TPR(x_{thr}) versus FPR(x_{thr}) with x_{thr} as the varying parameter. Our confidence in the ROC curve is thus directly dependent on our confidence in the estimation of f_0 and f_1. These two distributions are approximately Gaussian for our data, which is not surprising given the central limit theorem. We can thus use well-known expressions to determine the standard deviation of our estimates of the mean (μ) and standard deviation (σ) of f_0/f_1 based on n samples [39]:

$$\sigma_\mu = \frac{\sigma}{\sqrt{n}} \qquad \sigma_\sigma \approx \frac{\sigma}{\sqrt{2(n-1)}} \tag{32}$$

Both errors are approximately proportional to the inverse of $\sqrt{n}$. For $n = 4 \times 10^4$, both these errors are thus 200 times smaller than the standard deviation of the original distribution. This clearly shows that 4×10^4 samples result in a rather small confidence interval, thus validating our results.

To validate this theoretical analysis, we also carried out 10^5 tests for the case of 3 contacts. Then we plotted several ROC curves, each obtained using 10^4 samples, randomly selected from the 10^5 tests. We can see in Fig. 9 that the difference between the curves is reasonable, and it quantifies the uncertainty of our results.

REFERENCES

[1] 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 International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, jul 2012.

[2] J. Pratt, T. Koolen, T. de Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus, "Capturability-based analysis and control of legged locomotion, Part 2: Application to M2V2, a lower-body humanoid," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1117–1133, aug 2012.

[3] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture Point: A Step toward Humanoid Push Recovery," in *IEEE-RAS International Conference on Humanoid Robots*, 2006.

[4] P.-B. Wieber, "On the stability of walking systems," in *International Workshop on Humanoid and Human Friendly Robotics*, 2002.

[5] P.-B. Wieber, R. Tedrake, and S. Kuindersma, "Modeling and Control of Legged Robots," in *Springer Handbook of Robotics*, 2nd ed., B. Siciliano and K. Oussama, Eds., 2015, ch. 48.

[6] A. Del Prete, S. Tonneau, and N. Mansard, "Fast Algorithms to Test Robust Static Equilibrium for Legged Robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[7] D. Mansour, A. Micaelli, A. Escande, and P. Lemerle, "A computational approach for push recovery in case of multiple noncoplanar contacts," *IEEE-RAS International Conference on Humanoid Robots*, 2011.

[8] D. Mansour, A. Micaelli, and P. Lemerle, "Humanoid push recovery control in case of multiple non-coplanar contacts," in *IEEE/RSJ Intenational Conference on Intelligent Robots and Systems (IROS)*, 2013.

[9] P.-B. Wieber, "Viability and predictive control for safe locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[10] S. K. Yun and A. Goswami, "Momentum-based reactive stepping controller on level and non-level ground for humanoid robot push recovery," in *IEEE International Conference on Intelligent Robots and Systems*, 2011.

[11] O. E. Ramos and K. Hauser, "Generalizations of the Capture Point to Nonlinear Center of Mass Paths and Uneven Terrain," in *IEEE-RAS International Conference on Humanoid Robots*, 2015.

[12] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-Dimensional Bipedal Walking Control Based on Divergent Component of Motion," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 355–368, 2015.

[13] X. Xinjilefu, S. Feng, and C. G. Atkeson, "Center of mass estimator for humanoids and its application in modelling error compensation, fall detection and prevention," in *IEEE-RAS International Conference on Humanoid Robots*, 2015.

[14] S. Kalyanakrishnan and A. Goswami, "Learning To Predict Humanoid Fall," *International Journal of Humanoid Robotics*, vol. 08, no. 02, pp. 245–273, 2011.

[15] O. Höhn, J. Gačnik, and W. Gerth, "Detection and classification of posture instabilities of Bipedal robots," in *International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, (CLAWAR)*, 2005.

[16] R. Renner and S. Behnke, "Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[17] S.-k. Yun, A. Goswami, and Y. Sakagami, "Safe fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[18] K. Fujiwara, S. Kajita, K. Harada, K. Kaneko, M. Morisawa, F. Kanehiro, S. Nakaoka, and H. Hirukawa, "An Optimal planning of falling motions of a humanoid robot," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[19] J. Wang, E. C. Whitman, and M. Stilman, "Whole-body trajectory optimization for humanoid falling," in *American Control Conference (ACC)*, 2012.

[20] S. Ha and C. K. Liu, "Multiple Contact Planning for Minimizing Damage of Humanoid Falls," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[21] V. C. Kumar, S. Ha, and C. K. Liu, "Learning a Unified Control Policy for Safe Falling," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[22] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 794–807, 2008.

[23] Z. Qiu, A. Escande, A. Micaelli, and T. Robert, "A hierarchical framework for realizing dynamically-stable motions of humanoid robot in obstacle-cluttered environments," in *IEEE-RAS International Conference on Humanoid Robots*, 2012.

[24] K. Fukuda and A. Prodon, "Double Description Method Revisited," in *Combinatorics and Computer Science*, 1996.

[25] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-Optimal Control of Robotic Manipulators Along Specified Paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985.

[26] O. E. Ramos, N. Mansard, and P. Soueres, "Whole-body Motion Integrating the Capture Point in the Operational Space Inverse Dynamics Control," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2014.

[27] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota, and T. Isozumi, "Design of prototype humanoid robotics platform for HRP," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002, pp. 2431–2436.

[28] C. Mummolo, L. Mangialardi, and J. H. Kim, "Numerical Estimation of Balanced and Falling States for Constrained Legged Systems," *Journal of Nonlinear Science*, 2017.
[29] J. Carpentier, R. Budhiraja, and N. Mansard, "Learning Feasibility Constraints for Multicontact Locomotion of Legged Robots," in *Robotics, Science and Systems (RSS)*, 2017.
[30] A. Goswami, "Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.
[31] S. Rubrecht, V. Padois, P. Bidaud, and M. De Broissia, "Constraints Compliant Control : constraints compatibility and the displaced configuration approach," in *IEEE/RSJ Intenational Conference on Intelligent Robots and Systems (IROS)*, 2010.
[32] S. Tonneau, A. Del Prete, J. Pettre, C. Park, D. Manocha, and N. Mansard, "A fast and efficient acyclic contact planner for multiped robots," *IEEE Transaction on Robotics (under review)*, 2017.
[33] C. Brasseur, A. Sherikov, C. Collette, D. Dimitrov, and P.-B. Wieber, "A Robust Linear MPC Approach to Online Generation of 3D Biped Walking Motion," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2015.
[34] A. Sherikov, D. Dimitrov, and P.-b. Wieber, "Balancing a humanoid robot with a prioritized contact force distribution," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2015.
[35] A. Saccon, S. Traversaro, F. Nori, and H. Nijmeijer, "On Centroidal Dynamics and Integrability of Average Angular Velocity," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 943–950, 2017.
[36] J. Englsberger, C. Ott, M. a. Roa, A. Albu-Schäffer, and G. Hirzinger, "Bipedal walking control based on capture point dynamics," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2011.
[37] A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming- the explicit solution," *IEEE Transactions on Automatic Control*, vol. 47, no. 12, pp. 1974–1985, 2002.
[38] A. Del Prete, "Joint Position and Velocity Bounds in Discrete-Time Acceleration / Torque Control of Robot Manipulators," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, 2018.
[39] S. Ahn and J. Fessler, "Standard Errors of Mean, Variance, and Standard Deviation Estimators," *EECS Department, University of Michigan*, no. 2, pp. 2–3, 2003.