

# QP-based Adaptive-Gains Compliance Control in Humanoid Falls

Vincent Samy<sup>1</sup>, Karim Bouyarmane<sup>2</sup>, and Abderrahmane Kheddar<sup>1</sup>

**Abstract**—We address the problem of humanoid falling with a decoupled strategy consisting of a pre-impact and a post-impact stage. In the pre-impact stage, geometrical reasoning allows the robot to choose appropriate impact points in the surrounding environment and to adopt a posture to reach them while avoiding impact-singularities and preparing for the post-impact. The surrounding environment can be unstructured and may contain cluttered obstacles. The post-impact stage uses a quadratic program controller that adapts on-line the joint proportional-derivative (PD) gains to make the robot compliant –to absorb impact and post-impact dynamics, which lowers possible damage risks. This is done by a new approach incorporating the stiffness and damping gains directly as decision variables in the QP along with the usually-considered variables of joint accelerations and contact forces. Constraints of the QP prevent the motors from reaching their torque limits during the fall. Several experiments on the humanoid robot HRP-4 in a full-dynamics simulator are presented and discussed.

## I. INTRODUCTION

In order to effectively make use of humanoid robots in real-life applications such as daily home services<sup>1</sup>, large scale manufacturing<sup>2</sup>, or disaster response scenarios exemplified by the DARPA Robotics Challenge (DRC), it is mandatory to properly address the robot falling risk and to attenuate as much as possible the damage inherent to falling. It is indeed widely accepted that (i) even if the environment is well structured and even if we devote advanced strategies to walking, a humanoid robot will fall; and (ii) we are not able to list all the possible cases and situations where this will occur. A general common sense approach that accounts for the humanoid falling event would ideally operate as follows:

- devise strategies to avoid falling in the first place;
- if falling cannot be avoided in (a), or, for some reasons the robot must fall on purpose, then, if the robot is in servo-on, reduce as much as possible the damage resulting from the fall;
- when the two previous solutions are not applicable, i.e. if the robot is no more under control, it is better to simply resort to an extra shock absorbing system, such as an airbag, that can be triggered independently from the robot embedded control board.

In [1], we proposed a falling strategy for the case/step (b) above consisting of:

- A taxonomy to choose appropriate falling postures to adopt when falling is detected;
- Active reshaping, during which PD gains are high, to meet the impact in the best possible posture;
- Impact absorption by reducing the PD gains.

The above respective high and low PD gains values were manually ad-hoc tuned in [1]. The contribution of

<sup>1</sup>V. Samy and A. Kheddar are with CNRS - University of Montpellier LIRMM, 34000 Montpellier France [vincent.samy@lirmm.fr](mailto:vincent.samy@lirmm.fr)

<sup>2</sup>K. Bouyarmane is with University of Lorraine - INRIA - CNRS LORIA, 54600 Villers-lès-Nancy, France

<sup>1</sup>[www.projetromeo.com](http://www.projetromeo.com)

<sup>2</sup>[www.comanoid.eu](http://www.comanoid.eu)

the present paper is a method to tune them automatically in an adaptive way. Our novel idea consists in integrating the gain adaptation problem directly into the multi-objective QP formulation. This way, we can benefit from the on-line capabilities of the QP control approach –which has been widely adopted for controlling humanoid robots, and at the same time use the remaining “degrees-of-freedom” of the control for other tasks that can appear to be useful or necessary during falling, such as posture and CoM tasks as will be demonstrated later.

The rest of the paper is organized as follows. Section II reviews the related work in humanoid falling. Section III introduces the notation and hypotheses used throughout the paper. The two components of our approach are detailed in Sections IV and V. Section IV describes the pre-impact stage with the geometrical search of appropriate landing points and posture reshaping to prepare the impact. Section V deals with the post-impact stage detailing the joint motor PD gain computation inside the QP. Section VI presents simulation experiments in Gazebo on the humanoid robot HRP-4 to validate our approach, and Section VII concludes the paper with future work.

## II. RELATED WORK

We chose to focus on two main problems.

The first one is related to the strategy the robot should apply when falling in a cluttered environment. This kind of problem has been addressed in [2]. Based on inertia reshaping principles, they suggested three ways of modifying the direction of the fall. The concern is to avoid falling on/into a human.

The second problem we treat in this paper focuses on implicit damage reduction at the impact. This has been studied in [3], [4], [5], [6]. They proposed an off-line nonlinear method and an on-line solution to minimize the impact at landing for front and back falls. To prevent damaging the actuators, they are turned off just before the impact and turned on again right after. They also added an extra soft skin on the robot in order to absorb part of the shock.

In [7], [8] an on-line solution for front fall from a walking state is proposed. The idea is to track a CoM trajectory which aims at minimizing the impact.

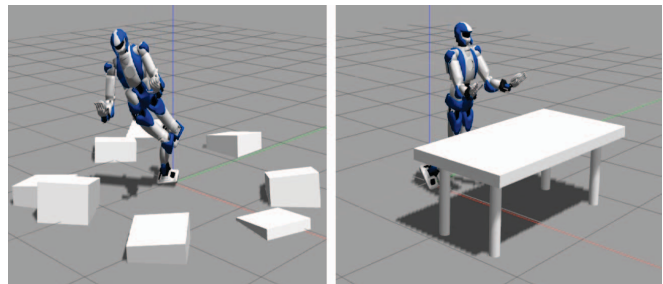


Fig. 1: Examples of falling in a cluttered environment.

Another method proposed in [9] consists in making the robot fall on its backpack that prevents the damage.

Finally, a tripod fall has also been considered in [10]. The idea comes from the simple observation that the earlier the impact occurs, the lower the kinetic energy is. So the method aims at breaking the fall as soon as possible by having the two hands and a foot touch the ground.

In very recent work [11], Ha and Liu presented an off-line strategy where an algorithm that finds a sequence of contacts minimizes the damage of a humanoid robot fall. [12] and [13] proposed a strategy based on an active cover.

In our previous work [1] we made a taxonomy of singular falls and proposed a simple fall strategy based on geometrical properties. We also tuned the PD gains of the motors to experimental values that allowed compliance at the impact.

Our contribution with respect to that previous work and to the listed state-of-the-art<sup>3</sup> is twofold: First, we extend the fall strategy to handle any fall direction in a cluttered environment, with more than just a posture task for falling on a flat floor as was the case in [1], see Fig 1. Secondly, we propose a novel method to automatically tune the PD gains within the whole-body QP controller instead of manually fixing experimentally drawn values as was the case in [1].

### III. TERMINOLOGY AND NOTATION

The mathematical notation we use is mostly the same as in [14] and [15]. Bold characters are vectors. If  $A$  and  $B$  are two spatial frames, then:

- ${}^B E_A$  denotes the rotation from  $A$  to  $B$ .
- ${}^B r_A$  denotes the position of the origin of  $B$  in  $A$ .
- $\mathbf{u}$  denotes a  $3 \times 1$  column vector in world coordinates.

In the notation  ${}^{r,b,p} X_0$ , the left-hand side superscripts and right-hand side subscript mean that the transformation is made from 0 to the point  $p$  of body (link)  $b$  of the robot  $r$  (i.e.  $p \in b \in r$ ). 0 is the index denoting the world.  $r$  is for robot and  $e$  for environment. Leaving left-hand side superscripts unspecified such as in the notation  $\mathbf{u}$  implicitly stands for  ${}^0 \mathbf{u}$ . Finally, right-hand side superscripts will eventually be used for complementary information.

### IV. GLOBAL FALLING STRATEGY

Fall control can be divided into four main parts: 1) Fall detection, 2) Environment analysis, 3) Pre-impact strategy execution, and 4) Post-impact strategy execution.

In **step 1**) a fall detection system must be constantly running in parallel to the performed tasks as a background process. The system should be able to stop the execution of the current tasks and switch to the pre-impact strategy execution whenever necessary. Note that this step might also include a fall recovery mode if possible.

In **step 2**) the robot performs an analysis of the situation (we exclude having humans or valuable items in the surroundings) in order to process useful information such as estimating the current pose of the robot and building a map of the surrounding's planar surfaces. Step 2) is out of this paper's scope. We shall consider it as a black-box module and assume that the environment, the robot state, and the available environment falling surfaces are known. This is a plausible assumption considering the advances made recently in SLAM technology [16].

<sup>3</sup><https://icra2016wsfallingrobots.wordpress.com/program/>

When the fall is detected, the controller goes through different states at each iteration loop in **step 3**), as follows:

- estimate the fall direction,
- search landing points,
- update falling tasks,

This step is detailed in subsections IV-A, IV-B, and IV-C.

At **step 4**), the robot has touched down. This step is considered independently from the previous steps, although the same whole-body controller is essentially used for both, as detailed in Section V. Additionally, the one we propose for this step will ensure an active compliance of the actuators/joints after the impact has occurred.

In this work, since the robot is under multiple tasks and constraints, we rely on a multi-objective weighted quadratic-program-based (QP) controller. The highest priority level is the QP constraints that must be satisfied without compromise:

- Joint limits, joint velocity limits, and torque limits
- Self-collision avoidance,
- The equation of motion and the physics model.

The second priority level (lowest level) are the tasks that are formulated as *set-point* tasks and combined in a weighted sum to form the objective function of the QP. A set-point task is defined as the minimization of the following quadratic cost:

$$E^{sp} = \frac{1}{2} \|S_M(k_p \mathbf{g} + k_v \dot{\mathbf{g}} + J_g \ddot{\mathbf{q}} + \dot{J}_g \dot{\mathbf{q}})\|^2, \quad (1)$$

where  $S_M$  is a selection matrix,  $k_p$  and  $k_v$  are proportional and damping task gains (not to be confused with the low-level joint PD gains that will be computed in Section V),  $\mathbf{g}$  is the task error and  $J_g$  is its Jacobian. More details on the controller can be found in [14].

In the following three subsections, we give further details on the three states described in **step 3**).

#### A. Direction of the fall

The fall direction is derived from the center of mass (CoM) trajectory. Let  ${}^{pc} \mathbf{r}_0^t$  be the CoM projected on the plane  $\mathcal{D}_0$  normal to the gravity and passing through a point on the ground, at time  $t$ . At each time step the fall direction is computed as

$$\mathbf{d}_f = \frac{{}^{pc} \mathbf{r}_0^t - {}^{pc} \mathbf{r}_0^{t_d}}{\|{}^{pc} \mathbf{r}_0^t - {}^{pc} \mathbf{r}_0^{t_d}\|}, \quad (2)$$

where  $t_d$  is the fall detection time and  $t > t_d$ .

#### B. Search of landing points

In order to choose the landing/impact points, we first need to know the potential impact surfaces. For a humanoid robot, the impact points are the hands, feet and knees [1]. We also assume here that a SLAM routine coupled with appropriate segmentation algorithms can return the available planar surfaces in the environment, as in [16]. In simulation however this information is readily available.

To lower damage risks resulting from the fall, we need to decide where to locate the impact points. These should be *reachable*, meaning that the robot can change its configuration in order to meet the desired landing spots during the falling time.

We approximate the robot as rigid stick that is falling, see Fig. 2 (green model on the figure). This stick lives in the

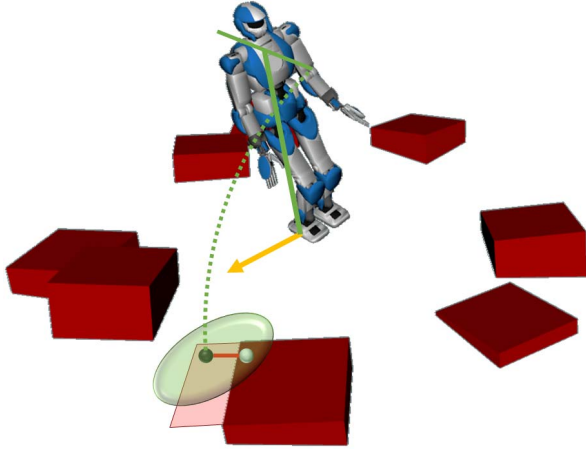


Fig. 2: Illustration of the search of possible impact points. The yellow arrow is the fall direction, the green lines represent a simplified stick model. The dotted arc is its trajectory. The transparent red plane is the plane where an impact surface exists. The transparent green ellipsoid is a gross representation of the polyhedron representing the arm's reachable workspace. Black and white points are the MPIP and BIP respectively. The red line represents the minimum distance between MPIP and BIP.

plane defined by a contact point (if any, or a projection of the nearest point from feet), the fall direction vector and the gravity vector. The length of the stick is set to the distance between the latter point and the middle of the two shoulders. Both the plane of motion of the stick and its length are adjusted at each time step. The trajectory of this stick in the defined plane is a 2D circle. The shoulders' trajectory are directly computed from it and the desired whole-body posture of the robot is then generated aiming for the hands to be on their respective shoulders' trajectory.

Finally, we compute all the intersections between shoulders' trajectory and the planes of the surfaces returned by **step 2**). We call these points *most probable impact points (MPIP)* hereafter. Fig. 2 represents one such MPIP as a black point. These points may or may not be on the environment surfaces (in the example of Fig. 2, the MPIP does not belong to the environment), this is why we also need to compute for each MPIP its closest point belonging to its respective environment surface. These closest points on the environment surfaces are called *Best Impact point (BIP)* (Fig 2 represents the BIP corresponding to the MPIP as a white point).

We now need to make a choice between the different available BIP. First, the arms' workspace gives two polyhedra which are split in two by the coronal plane, leading to one polyhedron for reachable front fall points and one for reachable back fall points. We also compute the centroid of each polyhedron. Note that these are calculated off-line only once and are associated with the geometric model of the robot. Placing the centroid on the MPIP, the BIP is selected if the segment of line between the MPIP and the BIP is inside the given polyhedron.

In case more than one BIP satisfies the condition, then the highest BIP (highest vertical coordinate) is chosen because the impact happens sooner and less potential energy is converted into kinetic energy before the impact. In case none

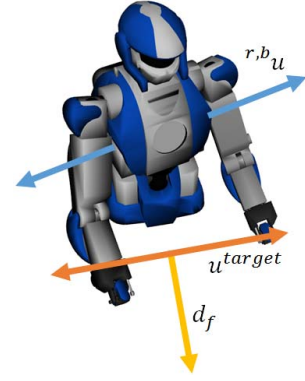


Fig. 3: The four possibilities for the vector orientation task. The blue vectors are the two possible body vector  ${}^{r,b}\mathbf{u}$ . The yellow vector is the fall direction and the two orange vectors are the possible targets  $\mathbf{u}^{\text{target}}$ .

of the points are inside the polyhedron, the BIP having the minimum distance to its respective MPIP is chosen.

### C. Reshaping tasks

To make the robot directly face the impact environment (front fall) or directly oppose it (back fall), since those two falls are the safest falls [1], we propose to use a vector orientation task which aims to align two vectors, one of which is linked to the torso and the other to the environment. A posture task is also included to help avoiding singular falls as defined in [1] and to bend the knees to lower the CoM. Finally, end-effector position tasks are used to reach the desired impact points. All of these task are run and their targets updated at each control loop. To implement a new set-point task (1), the task error  $\mathbf{g}$ , the task Jacobian  $J_g$  and the time-derivative of the task Jacobian  $\dot{J}_g$  are needed. We describe these derivations in the next subsections.

1) *Vector orientation task*: Let  $\mathbf{u}^{\text{target}}$  be the desired goal unit vector and  ${}^{r,b}\mathbf{u}$  a unit vector in robot body coordinates. The task error is given by:

$$\begin{aligned} \mathbf{g}_{vo} &= \mathbf{u}^{\text{target}} - {}^0E_{r,b} {}^{r,b}\mathbf{u}, \\ &= \mathbf{u}^{\text{target}} - {}^0\mathbf{u}, \end{aligned} \quad (3)$$

where  ${}^0E_{r,b}$  is the rotation from the body  $b$  to the world. As the target vector is considered as fixed in the world, only the time-derivative of the robot vector is considered:

$$\begin{aligned} {}^0\dot{\mathbf{u}} &= {}^0E_{r,b} [{}^{r,b}\boldsymbol{\omega} \times {}^{r,b}\mathbf{u}], \\ &= -{}^0E_{r,b} ({}^{r,b}\mathbf{u} \times) {}^{r,b} J_{r,b}^{\text{ang}} \dot{\mathbf{q}}, \\ &= J_{g_{vo}} \dot{\mathbf{q}}. \end{aligned} \quad (4)$$

Here,  ${}^{r,b}\boldsymbol{\omega}$  is the angular velocity of the body in body coordinates and  ${}^{r,b}J_{r,b}^{\text{ang}}$  is the angular part of the body Jacobian in body coordinates. Differentiating one more time, the time-derivative of the task Jacobian is then:

$$\dot{J}_{g_{vo}} = {}^0E_{r,b} [{}^{r,b}\boldsymbol{\omega} \times ({}^{r,b}\boldsymbol{\omega} \times {}^{r,b}\mathbf{u}) + {}^{r,b}\mathbf{a}^{\text{vp,ang}} \times {}^{r,b}\mathbf{u}], \quad (5)$$

where  ${}^{r,b}\mathbf{a}^{\text{vp,ang}} = {}^{r,b}J_{r,b}^{\text{ang}} \dot{\mathbf{q}}$  is the angular part of velocity-product of the acceleration in body coordinates [15].

The targeted vector is set so that  $\mathbf{u}^{\text{target}} \in \mathcal{D}_0$  and  $\mathbf{u}^{\text{target}} \cdot \mathbf{d}_f = 0$  (Fig. 3).  ${}^{r,b}\mathbf{u}$  is chosen perpendicular to the torso sagittal plane in the torso coordinates. There are four

possible solutions so both vectors must be chosen depending on whether a front fall or a back fall is desired.

2) *Relative distance task*: Ideally, we would like that multiple impacts occur all at the same time, but this is difficult to achieve in practice because it requires estimation of the exact impact time. A solution is to manipulate the distance between the desired environment impact surfaces and the robot impacting bodies so that the relative error of the distances between two pairs of surface-impacting bodies is zero. One of the advantages of this task is that it handles different heights of surfaces. We remind here that the considered surfaces are planar so the time-derivatives of their normals are zero. Let  ${}^{r,b_1,p_1}\mathbf{r}_0$  be the closest point of a body  $b_1$  to a surface  $s_1$  and  ${}^{r,b_2,p_2}\mathbf{r}_0$  the closest point of a body  $b_2$  to a surface  $s_2$ . Let  ${}^{e,s_1,p_1}\mathbf{r}_0$  and  ${}^{e,s_2,p_2}\mathbf{r}_0$  be points on  $s_1$  and  $s_2$  respectively. The distance of a pair of impact body and surface is:

$$d_i = \|{}^{r,b_i,p_i}\mathbf{r}_0 - {}^{e,s_i,p_i}\mathbf{r}_0\|, \quad i = 1, 2. \quad (6)$$

Here, we do not want to consider the minimal distance but rather a distance along an axis, which is more useful in our application. The task is designed to modify the distance between robot bodies and surface planes, so the distance along the normal of a plane is more relevant. This method controls only the motion along the normal of the plane, while the motion along the plane itself is left free and will be handled by a position task for reaching the desired impact points.

Let now  $\mathbf{u}_1$  and  $\mathbf{u}_2$  be unit vectors linked to  $s_1$  and  $s_2$  respectively. The task error is:

$$g_{rd} = d_1 \cdot \mathbf{u}_1 - d_2 \cdot \mathbf{u}_2. \quad (7)$$

The surfaces  $s_1$  and  $s_2$  are considered fixed so the time-derivatives of  $\mathbf{u}_i$  and  ${}^{e,s_i,p_i}\mathbf{r}_0$  is zero ( $i = 1, 2$ ). Note that if this assumption is false, then it means that the robot would fall on a moving environment. It is possible to adapt the tasks to handle such cases but we will not consider them here. The time-derivative of  $g$  is given by:

$$\begin{aligned} \dot{g}_{rd} &= \mathbf{v}_1 \cdot \mathbf{u}_1 - \mathbf{v}_2 \cdot \mathbf{u}_2, \\ &= (\mathbf{u}_1^T J_{r,b_1,p_1}^{\text{lin}} - \mathbf{u}_2^T J_{r,b_2,p_2}^{\text{lin}}) \dot{\mathbf{q}}, \\ &= J_{g_{rd}} \dot{\mathbf{q}}, \end{aligned} \quad (8)$$

where  $J_{b_i,p_i}^{\text{lin}}$  is the linear part of the body Jacobian of point  $p_i$  associated to body  $b_i$  ( $i = 1, 2$ ). The Jacobian time-derivative is:

$$\dot{J}_{g_{rd}} = \mathbf{u}_1^T \dot{J}_{b_1,p_1}^{\text{lin}} - \mathbf{u}_2^T \dot{J}_{b_2,p_2}^{\text{lin}}. \quad (9)$$

3) *End-effector position task*: The end-effector position task is a common task [14]. The points on the hands to control are the closest points to their respective chosen BIP. We also mention that the task should be written in the surface frame so that only the  $x$  and  $y$  coordinates are controlled by the task. The  $z$ -coordinate (normal to the surface) is handled by the relative-distance task above.

## V. POST-IMPACT STRATEGY

The pre-impact process described in section IV shapes the robot into a relatively ‘‘compliant’’ posture just before the impact. The impact is produced whenever the feet, knees or hands are about to touch down. From that instant, the controller behaves as an active compliance for lowering the damage, using a single QP whole-body controller.

In position-controlled humanoids, the low-level actuator controller consists in a proportional-derivative (PD), which leads to the simplified governing equation:

$$H\ddot{\mathbf{q}} + \mathbf{C} - J^T G \boldsymbol{\lambda} = \boldsymbol{\tau} = \mathbf{K}e + \mathbf{B}\dot{e}, \quad (10)$$

where  $H \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{dof}}}$  is the robot inertia matrix,  $\mathbf{C} \in \mathbb{R}^{N_{\text{dof}} \times 1}$  the gravity and Coriolis vector,  $J \in \mathbb{R}^{6N_c \times N_{\text{dof}}}$  the contact Jacobian,  $G \in \mathbb{R}^{6N_c \times N_c N_g}$  the matrix of friction cone generators, and  $\boldsymbol{\tau}$  the generalized forces (comprising the actuation torques for the actuated joints and zero entries for the non-actuated ones). The parameters  $\mathbf{K} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{dof}}}$  and  $\mathbf{B} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{dof}}}$  are the diagonal matrices of PD gains,  $e = \mathbf{q}_{\text{ref}} - \mathbf{q}$  and  $\dot{e} = \dot{\mathbf{q}}_{\text{ref}} - \dot{\mathbf{q}}$  are respectively the errors in joint position and velocity.  $\mathbf{q}_{\text{ref}}$  is set to the current configuration just before the impact and  $\dot{\mathbf{q}}_{\text{ref}}$  is set to zero. Note that in the case of joints without motors (e.g. the free-floating base) the corresponding entries in the diagonals of  $\mathbf{K}$  and  $\mathbf{B}$  are zeros. We denote  $\mathbf{K}$  and  $\mathbf{B}$  the vectors containing the diagonal entries of  $\mathbf{K}$  and  $\mathbf{B}$  respectively, i.e.  $\mathbf{K} = \text{diag}(\mathbf{K})$  and  $\mathbf{B} = \text{diag}(\mathbf{B})$ .  $N_{\text{dof}}$ ,  $N_c$  and  $N_g$  are respectively the number of degrees of freedom (dof) of the system, the number of contact points and the number of generators of the linearized friction cones. We also define  $N_m$  as the number of motors.  $\mathbf{K}$  and  $\mathbf{B}$  have constant values that encode the default high stiffness behavior of the motors. These values are generally very high to make the motors track the reference values as fast as possible accounting for perturbations, inertia –and more general dynamics, while avoiding overshooting. In order to comply, we need to relax and adapt these values.

Our novel idea is to use a multi-objective QP formulation in the  $\mathbf{X} = (\dot{\mathbf{q}}, \boldsymbol{\lambda}, \mathbf{K}, \mathbf{B})$  decision vector.

First, to handle the impact/contact, a constraint is added so that at the contact points, the velocity is zero. This condition is realized with the following constraints [17]:

$$S_M \left( \frac{\mathbf{v} - \bar{\mathbf{v}}}{\Delta T} \leq J\ddot{\mathbf{q}} + J\dot{\mathbf{q}} \leq \frac{\bar{\mathbf{v}} - \mathbf{v}}{\Delta T} \right), \quad (11)$$

where  $S_M$  is a  $n \times 6$  ( $n \leq 6$ ) selection matrix,  $\mathbf{v}$  and  $\bar{\mathbf{v}}$  are the minimal and maximal body velocity.

The primary objective of a compliant behavior is the motor’s constraints. They are modeled as box torque constraints and added to the QP as follows

$$\underline{\boldsymbol{\tau}} \leq \mathbf{K}e - \mathbf{B}\dot{e} \leq \bar{\boldsymbol{\tau}}. \quad (12)$$

The other box constraints are the bounds over the parameters in  $\mathbf{X}$ :

$$\begin{cases} \underline{\dot{\mathbf{q}}} \leq \dot{\mathbf{q}} \leq \bar{\dot{\mathbf{q}}} \\ \boldsymbol{\lambda} \geq \mathbf{0} \\ \mathbf{K} \geq \mathbf{0} \\ \mathbf{B} \geq \mathbf{0} \end{cases}. \quad (13)$$

*Important note*: in post-impact, constraints on joint limits and velocity limits are purposely not inserted as constraints in the QP. The reason for this is that we have no control over the impact behavior. Indeed, the impact is imposed to the robot in a very limited time. If it is large enough, the generated velocity would make the QP fail to find a feasible solution so the robot would remain stiff. Thus, the main advantage of not taking limits as constraints is that the robot will always comply until it has fully absorbed the post-impact dynamics or until it reaches a mechanical stop

(joint limit). On the opposite, in case the impact is not large enough, nothing guarantees that the robot will not reach a joint limit. In order to ensure that the joints are kept inside their limits, a basic strategy would be to give high weight and stiffness to a posture task instead. This amounts to changing the ‘priority’ level, i.e. shifting the joint limit constraints from the constraint set to the cost function in the QP.

Finally, the QP writes as follow:

$$\begin{cases} \min_{\ddot{\mathbf{q}}, \boldsymbol{\lambda}, \mathbf{K}, \mathbf{B}} & \sum_k \omega_k^{sp} E_k^{sp} + \omega_\lambda \|\boldsymbol{\lambda}\|^2 + \omega_G (\|\mathbf{K}\|^2 + \|\mathbf{B}\|^2), \\ \text{s.t.} & (10), (11), (12), (13) \end{cases} \quad (14)$$

where  $k$  is an index over the tasks (posture, CoM). The number of parameters is equal to  $\dim(\mathbf{X}) = N_X = N_{\text{dof}} + N_c N_g + 2N_m$ , which is almost three times the number of variables of the more usual form of the QP used for general-purpose control (i.e. without  $\mathbf{K}$  and  $\mathbf{B}$ ). In order to improve the performance we chose to restrain the gain adaptation only to a selected set of joints directly involved in impact absorption. We propose to select all the motors in the kinematic chain between the end-effector contact points and the root of the kinematic chain of the robot. For example, if a contact is on the hand (actually the wrist on the HRP-4 robot), then the motors of the elbow and the shoulder are retained.

Once the joints are selected, we just need to extract their corresponding lines in the matrices  $H$  and  $J^T G$  and in the vectors  $C$  and  $\tau$ . The other joints need also to be consistent with the dynamics and the torque limits so a new constraint is added to the QP. The constraints (10) and (12) are removed and the following constraints are added to the QP (14):

$$\begin{cases} H_S \ddot{\mathbf{q}} + \mathbf{C}_S - (J^T G)_S \boldsymbol{\lambda} = \mathbf{K} \mathbf{e}_S - \mathbf{B} \dot{\mathbf{e}}_S \\ \quad \quad \quad \tau_S \leq \mathbf{K} \mathbf{e}_S - \mathbf{B} \dot{\mathbf{e}}_S \leq \bar{\tau}_S \\ \tau_{NS} \leq H_{NS} \ddot{\mathbf{q}} + \mathbf{C}_{NS} - (J^T G)_{NS} \boldsymbol{\lambda} \leq \bar{\tau}_{NS} \end{cases} \quad (15)$$

with the subscript  $S$  (resp.  $NS$ ) designating the matrix/vector of selected (resp. non-selected) rows.

Note that both the pre-impact and post-impact stages can be performed. As all bodies are not impacting at the same time (in a front fall the knees generally impact way before the hands), this QP form allows to perform both pre-impact and post-impact in parallel. (At knees’ impact, the legs are set to the complying behavior whereas the upper part of the robot continues its pre-impact stage).

## VI. SIMULATIONS

To demonstrate the capabilities of the adaptive-gain QP, we performed several falling simulations of the HRP-4 robot in the Gazebo simulator (see companion video).

We focus in this section on the very first experiment consisting in dropping the robot from a given height (1m) and letting it land on its feet (at impact time  $t \simeq 0.45\text{s}$  with a velocity of 4.43m/s). Four methods are compared 1) keeping the robot’s stiff initial gains, 2) using predefined static gains (as in [1]), 3) using zero gains (shutting down the robot, in servo-off mode) and finally 4) adaptive gains (our proposed method). This experiment illustrates the post-impact gain adaptation strategy part of the paper. The pre-impact geometric reshaping part is illustrated along with the gain adaptation in all the other experiments of the video.

In order to back up our claim that the adaptive QP complies with the post-impact dynamics and lowers the risk

of damaging the robot, we chose to look at two qualifiers: (i) the IMU acceleration (in the waist) (Fig. 6a), and (ii) the joint positions (Fig. 6b). As the floating-base (waist) acceleration is proportional to the applied external forces, and as there are many contacts, we found the IMU acceleration to be a good indicator of how much total impact/contact force is applied on the structure. We use the IMU acceleration as damage quantification comparison quantity: the less acceleration there is, the better and the safer for the robot.

Let us first analyse the data from our proposed approach alone (Figs. 5). We can see that the damping coefficient increase very fast until 0.6s. This is mostly due to the fact that right after the impact, the error is almost null whereas the velocity is high, hence the solver is mostly using the damping gains  $B$  (Fig. 5). To understand the high variation of the damping gain around 0.6s (from 60 to 0Nms/rad), we note in Fig. 4 that around 0.6s the torque’s sign (minus) is unchanged. At this stage, the joint velocity is switching sign (from plus to minus). Considering the eq.  $\tau = \mathbf{K} \mathbf{e} + \mathbf{B} \dot{\mathbf{e}}$  we can see that in order to have a negative torque with a positive velocity error ( $\dot{\mathbf{e}} = 0 - \dot{\mathbf{q}}$ ) we need  $B < 0$ . But the constraint  $B \geq 0$  enforces the non-negativity of the damping coefficient resulting in a zero value for it.

Fig. 6b shows that using the predefined static fixed PD gains or turning off the motors could be extremely risky since joint limits are reached fast. On the other hand, keeping the initial (stiff) gains does not make the robot reach the mechanical stops but leads to very high jerk and IMU acceleration (6a) at the impact, which is a prediction of a high impact force. The proposed adaptive QP approach avoids all these issues. It has a low jerk and a low acceleration profile while still staying under the joint limit and not reaching any mechanical stop at the joints. Fig. 4 shows that the adaptive QP keeps the torques under their limits.

## VII. CONCLUSION AND FUTURE WORK

We proposed an original way of addressing falls in a cluttered environment. First, an active reshaping fall strategy prepares the robot to the impact from the moment the fall is detected and up to just before the impact occurs. Then, during the post-impact stage, a QP controller allows the robot to become compliant in order to absorb the impact energy while satisfying its structural constraints.

In order to implement this strategy on a real robot, two modules are necessary and were assumed as available black-

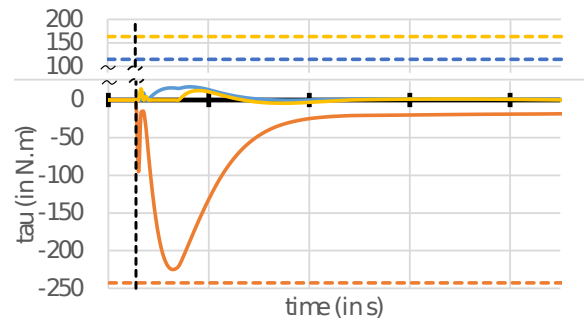


Fig. 4: Evolution of the torque for the three right leg pitch joints (hip, knee, ankle) resulting from our adaptive QP method. The dashed vertical line is the impact time. Dashed horizontal lines are torque limits.

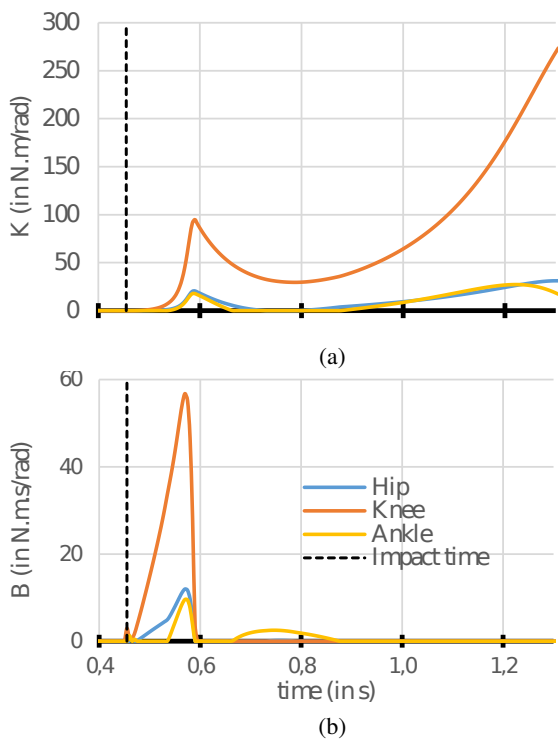


Fig. 5: Evolution of: (5a) the stiffness, and (5b) damping gains for the right leg pitch joints resulting from our QP adaptive method. The dashed line is the impact time.

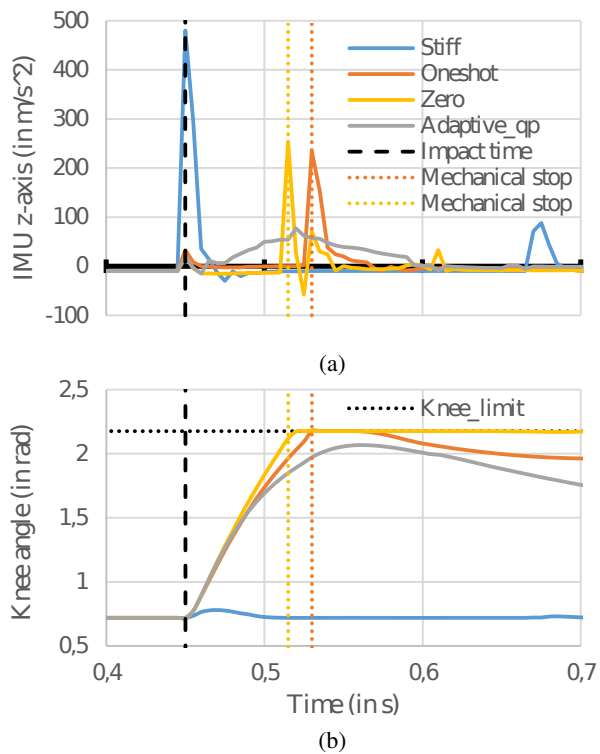


Fig. 6: (6a)  $z$ -axis of the IMU; (6b) right knee position. Four methods are represented. Stiff: default gains, Oneshot: fixed gains as in [1], Zero: zero gains, Adaptive.qp: adaptive gains. The black dotted line is the joint limit of the knee and the colored dotted lines represent mechanical stops (joint limits).

boxes: robot state estimation and landing surfaces candidates computation, both can be provided by SLAM in future work.

For now, sliding contacts are not perfectly handled in the QP. This is a challenging problem that we are currently working on for general multi-contact planning and control purposes. A temporary solution we implemented was to release the tangent space dof of the contacts to allow sliding.

Finally, fall detection itself needs to be improved. Many methods have been suggested, but all of them fail in several cases. In real situation, falling extends beyond what the current state-of-the-art can detect. For example, falling does not necessarily restrict to the notion of loss of balance because the latter may be dictated by a task to achieve. In all generality, it should be thought of as the loss of task-based controllability, but this novel concept is out of this paper's scope and needs to be researched as a new direction.

## REFERENCES

- [1] V. Samy and A. Kheddar, "Falls control using posture reshaping and active compliance," in *IEEE-RAS Int. Conf. on Humanoids*, 2015, pp. 908–913.
- [2] A. Goswami, S.-k. Yun, U. Nagarajan, S.-H. Lee, K. Yin, and S. Kalyanakrishnan, "Direction-changing fall control of humanoid robots: theory and experiments," *Autonomous Robots*, vol. 36, no. 3, pp. 199–223, 2014.
- [3] K. Fujiwara, F. Kanehiro, S. Kajita, K. Kaneko, K. Yokoi, and H. Hirukawa, "UKEMI: falling motion control to minimize damage to biped humanoid robot," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002, pp. 2521–2526.
- [4] K. Fujiwara, F. Kanehiro, S. Kajita, and H. Hirukawa, "Safe knee landing of a human-size humanoid robot while falling forward," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2004, pp. 503–508.
- [5] K. Fujiwara, S. Kajita, K. Harada, K. Kaneko, M. Morisawa, F. Kanehiro, S. Nakaoka, and H. Hirukawa, "Towards an optimal falling motion for a humanoid robot," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2006, pp. 524–529.
- [6] —, "An optimal planning of falling motions of a humanoid robot," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007, pp. 456–462.
- [7] K. Ogata, K. Terada, and Y. Kuniyoshi, "Falling motion control for humanoid robots while walking," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2007, pp. 306–311.
- [8] —, "Real-time selection and generation of fall damage reduction actions for humanoid robots," in *IEEE-RAS Int. Conf. on Humanoids*, 2008, pp. 233–238.
- [9] S.-H. Lee and A. Goswami, "Fall on backpack: Damage minimization of humanoid robots by falling on targeted body segments," *ASME J. of Computational and Nonlinear Dynamics*, vol. 8, no. 2, pp. 1–10, 2013.
- [10] S.-k. Yun and A. Goswami, "Tripod fall: Concept and experiments of a novel approach to humanoid robot fall damage reduction," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 2799–2805.
- [11] S. Ha and C. K. Liu, "Multiple contact planning for minimizing damage of humanoid falls," in *IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems*. IEEE, 2015, pp. 2761–2767.
- [12] J. Lee, W. Choi, D. Kanoulas, R. Subburaman, D. G. Caldwell, and N. G. Tsagarakis, "An active compliant impact protection system for humanoids: Application to walk-man hands," in *IEEE-RAS Int. Conf. on Humanoids*, 2016.
- [13] S. Kajita, R. Cisneros Limon, M. Benallegue, T. Sakaguchi, S. Nakaoka, M. Morisawa, and K. F. Kaneko, Kenji, "Impact acceleration of falling humanoid robot with an airbag," in *IEEE-RAS Int. Conf. on Humanoids*, 2016.
- [14] J. Vaillant, A. Kheddar, H. Audren, F. Keith, S. Brossette, A. Escande, K. Bouyarmane, K. Kaneko, M. Morisawa, P. Gergondet, E. Yoshida, S. Kajita, and F. Kanehiro, "Multi-contact vertical ladder climbing with an HRP-2 humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 561–580, 2016.
- [15] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.
- [16] R. F. Salas-Moreno, B. Glocken, P. H. J. Kelly, and A. J. Davison, "Dense planar slam," in *2014 IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR)*, Sept 2014, pp. 157–164.
- [17] J. Vaillant, K. Bouyarmane, and A. Kheddar, "Multi-character physical and behavioral interactions controller," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2016.