Noname manuscript No.
(will be inserted by the editor)

Direction-Changing Fall Control of Humanoid Robots: Theory and Experiments

Ambarish Goswami · Seung-kook Yun · Umashankar Nagarajan · Sung-Hee Lee · KangKang Yin · and Shivaram Kalyanakrishnan

Received: date / Accepted: date

Abstract Humanoid robots are expected to share human environments in the future and it is important to ensure the safety of their operation. A serious threat to safety is the fall of such robots, which can seriously damage the robot itself as well as objects in its surrounding. Although fall is a rare event in the life of a humanoid robot, the robot must be equipped with a robust fall strategy since the consequences of fall can be catastrophic.

In this paper we present a strategy to change the default fall direction of a robot, *during the fall*. By changing the fall direction the robot may avoid falling on a delicate object or on a person. Our approach is based on the key observation that the toppling motion of a robot necessarily occurs at an edge of its support area. To modify the fall direction the robot needs to change the position and orientation of this edge vis-a-vis the prohibited directions. We achieve this through intelligent stepping as soon as the fall is predicted. We compute the optimal stepping location which results in the safest fall. Additional improvement to the fall controller is achieved through *inertia shaping*, which is a principled approach aimed at manipulating the robot's centroidal inertia, thereby indirectly controlling its fall direction.

We describe the theory behind this approach and demonstrate our results through simulation and experiments of the Aldebaran NAO H25 robot. To our knowledge, this is the first implementation of a controller that attempts to change the fall direction of a humanoid robot.

A. Goswami
Principal Scientist
Honda Research Institute USA
Mountain View, CA 94043, U.S.A.
E-mail: agoswami@honda-ri.com

S.-K. Yun
Senior Software Engineer
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
E-mail: seungkook.yun@gmail.com

U. Nagarajan
Postdoctoral Associate
Disney Research Pittsburgh
Pittsburgh PA 15213 USA
E-mail: umashankar@disneyresearch.com

S.-H. Lee
Assistant Professor
Graduate School of Culture Technology
Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, South Korea
E-mail: leesunghee@gmail.com

K. Yin
Assistant Professor
School of Computing
National University of Singapore (NUS)
Singapore 117417
E-mail: kkyin@comp.nus.edu.sg

S. Kalyanakrishnan
Scientist
Yahoo! Labs Bangalore
Bengaluru India 560071
E-mail: shivaram@cs.utexas.edu

1 Introduction

Safety is a primary concern that must be addressed before humanoid robots can freely exist in interactive human surrounding. Although the loss of balance and fall are rare in typical controlled environments, it will be inevitable in physically interactive environments. Out of a number of possible situations where safety becomes

an issue, one that involves a fall is particularly worrisome. Fall from an upright posture can cause damage to the robot itself, to delicate and expensive objects in the surrounding or can inflict injury to a human being. Regardless of the substantial progress in humanoid robot balance control strategies, the possibility of fall remains real, even unavoidable. Yet, only a few comprehensive studies of humanoid fall (encompassing fall avoidance, prediction, and control) have been undertaken in the literature.

A humanoid fall may be caused due to unexpected or excessive external forces, unusual or unknown slipperiness, slope or profile of the ground, causing the robot to slip, trip or topple. In these cases the disturbances that threaten balance are larger than what the balance controller can handle. Fall can also result from actuator, power or communication failure where the balance controller is partially or fully incapacitated. In this paper we consider only those situations in which the motor power is retained such that the robot can execute a prescribed control strategy.

One can ignore the possibility of a fall and wishfully hope that its effects will not be serious. However, failure studies, such as in car crash, have taught us against following this instinct. In fact, planning and simulation of failure situations can have enormous benefits, including system design improvements, and support for user safety and confidence. With this philosophy we closely focus our attention to the phenomenon of humanoid fall and attempt to develop practical control strategies to deal with this undesired and traumatic failure event.

A controller dealing with an accidental fall may have two primary and distinctly different goals: a) self-damage minimization and b) minimization of damage to others. When a fall occurs in an open space, a self-damage minimization strategy can reduce the harmful effects of the ground impact. If, however, the falling robot can damage nearby objects or injure persons, the primary objective would be to prevent this. The current paper reports a control strategy for changing the default fall direction of the robot so that it avoids contact with surrounding objects or people as a mean of minimizing damage to others. Recently, Wilken et al. have reported a third possible goal of a fall controller, that of a deliberate fall of a humanoid soccer goalie [40]. This is the case of a strategic fall.

Time is at a premium during the occurrence of a fall; a single rigid body model of a full sized humanoid indicates that a fall from the vertical upright stationary configuration due to a mild push takes about 800-900 ms [37]. In many situations the time to fall can be significantly shorter, and there is no opportunity for elaborate planning or time-consuming control. Yet, through simulation and experiments we are able to demonstrate that meaningful modification to the default fall behavior can be achieved in a very short time and damage to the environment can be avoided.

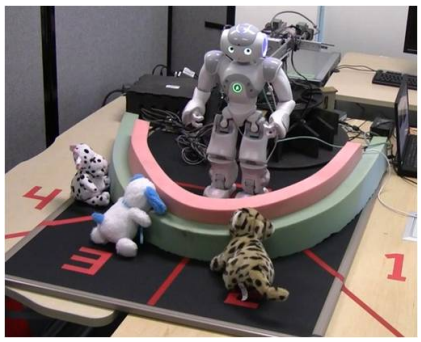
Let us clarify that *a fall controller is not a balance controller*. A fall controller complements, and does not replace, a balance controller. Further, *a fall controller is not a push-recovery controller*. A push-recovery controller is essentially an extended balance controller, which specifically deals with external disturbances of larger magnitude when the robot must take a step in order to regain balance ([28, 36, 41]). The fall controller is activated only when the default balance controller or the push recovery controller has failed to stabilize the robot.

We have reported our earlier work on humanoid fall direction change in [19, 26, 42, 43]. In the current paper we provide a comprehensive account of our work including generalization, extension and improvements. Also, we present results of hardware experiments of fall direction change control performed on an Aldebaran NAO H25 robot.

Examples of our current results are shown in Fig. 1, in which a "table top" humanoid robot (Aldebaran NAO H25 [14]) is surrounded by three dolls occupying three of the four 45° sectors of the semicircular area in front of the robot. The objective of the fall direction controller is to make the robot fall inside the empty sector given the same push by a linear actuator from behind. In different experimental trials, shown in Fig. 1(b-e), we rearrange the dolls in order to change the location of the empty sector. Using our fall controller, the robot successfully avoids hitting the dolls by falling into the empty sector.

2 Related Work

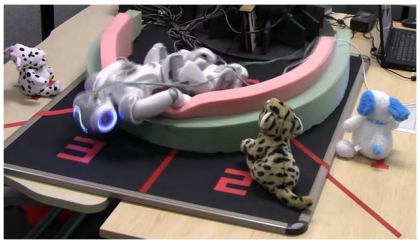
A number of recent papers reported on the damage minimization and prediction aspects of humanoid fall. In their exhaustive work, Fujiwara et al. ([9–13]) proposed martial arts type motion for damage reduction, computed optimal falling motions using minimum impact and angular momentum, and fabricated special hardware for fall damage study. Ogata et al. proposed two fall prediction methods based on abnormality detection and predicted Zero Moment Point (ZMP) [22, 27]. The robot improves fall prediction through experimental learning. Renner and Behnke [29] use model-based approach to detect external forces on the robot and Karssen and Wisse [21] use principal component analysis to predict fall. Hobbelen and Wisse [15] proposed Gait Sensitivity Norm which can be used as a fall detector. Following human movement based search proce-



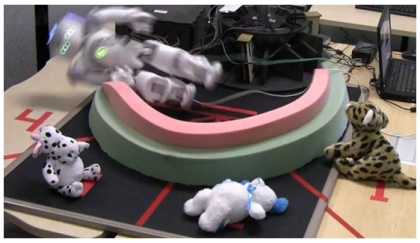
(a)

(b)

(c)



(d)



(e)

Fig. 1: (a) The Aldebaran NAO robot is pushed from behind by a linear actuator. Its front semicircle is divided into four equal sectors 45° each. Three dolls are placed in three arbitrary sectors while the fourth sector is empty. (b-e): In four trials, we rearrange the dolls in order to change the location of the empty sector relative to the robot. Under an identical push the robot successfully changes the fall direction in real time and falls into the empty sector to avoid hitting the dolls.

dure, Ruiz-del-Solar et al. implemented a low damage fall sequence for soccer robots [33, 34]. In [16] and [17], fall prediction and control are treated together using Gaussian mixture models and Hidden Markov model. Ishida et al. employed servo loop gain shift to reduce shock due to fall [18]. Kanoi et al. worked on fall detection from walk patterns [20]. Fall damage minimization is obviously of natural interest in human biomechanics [3–6].

2.1 Damage minimizing fall control strategies

Many of the previous works on damage minimizing fall focuses on implementing heuristics such as lowering the center of mass of a humanoid robot during fall. For example, Ruiz del Solar et al. [33] and Fujiwara et al. [13] used strategies taken from Japanese martial arts according to the direction of fall (forward, backward, sides) and each strategy is accompanied by a lowering of the CoM.

A few of the works listed below further reduced the damage from fall by designing specific trajectories for the CoM instead of just lowering it. However, these works were limited to forward fall only. Fujiwara et al. [9] added braking of the landing speed after lowering the CoM so that the falling robot reduces the impact velocity. The braking was achieved by stretching the body fast just before the robot hits the ground. Also, they reduced the feedback gain after braking in order to make the joints compliant. Ogata et al. used straight [22] and curvilinear [27] trajectories of the CoM, and both the trajectories virtually targeted the same two phases: lowering the CoM and re-stretching the body to reduce the vertical speed just before the impact. In hardware experiments, Fujiwara et al. [11] used a simplified humanoid robot and used optimization techniques to design the optimal joint trajectories for minimizing the impact velocity during a forward fall while maintaining all the constraints. Interestingly, the motions obtained were similar to the results from the previous works in that they also included the two similar phases.

Ruiz del Solar et al. [34], instead of minimizing the direct impact velocity, tried to minimize damage to each joint by reducing the axial force and torque resulting from the impact. They used motion-capture data to construct the base fall motions and hand-tuned them joint by joint.

In their study of intentional fall, Wilken et al. [40] adopted an inverse approach; instead of minimizing the fall damage, they first designed the fall motions and then changed the robot's structure to reduce the damage. Springs and flexible rubber struts were added to

the most damage-prone locations of the robot given the deliberate fall motion of a robot soccer goalie.

2.2 Biomechanical studies of human fall

At present, modeling, analysis and simulation of fall are active research topics in biomechanics. This research suggest that the body segment movements during a fall are not random and unpredictable, but involve directed efforts to land safely [32]. Fall strategies in humans can thus inspire biomimetic strategies for a humanoid robot.

Most fall strategies involving humans are meant for injury-minimization and they have two main objectives: 1) the reduction of impact velocity, and 2) the distribution of impact force to a larger contact area. Because fall strategy is a time-critical task, its success depends on how early a fall is predicted and a reactive action is initiated. Hence, the reaction time is an important factor for a fall [1].

Typical injury-minimization strategies of humans involve: 1) extending both arms to diffuse full or part of the impact energy [7]; 2) bending the elbows to break a fall with reduced impact force [7]; 3) ground touchdown with the knee to reduce the downward momentum of fall earlier than the arms [37]; 4) lowering of the CoM to reduce the vertical impact velocity and kinetic energy through energy absorption in the lower extremity muscles during descent [30, 31]; and 5) curling into a ball similar to athletes and martial artists.

Although biomechanics results can be very valuable for our study, we should also be aware of the limits to which they can be directly applied to humanoid robots. First, the biologically evolved human fall strategies probably operate under a learned and assigned value on different parts of the body with the high-value regions to be protected from an impact. Typically, the human instinct is to protect their head, which is considered high value, or the frontal face, perhaps to reduce the pain associated with a ground impact. This may not necessarily be applicable for a humanoid robot, for which it may be more worthwhile to protect an area of critical control circuitry. Further, due to marked differences in the material and actuation properties between humanoid and humans, the magnitude of impact forces for the two cases will be different.

It is remarkable to note the virtual absence of any literature on direction-changing fall in the field of biomechanics. This might be indicative of the fact that these strategies are not common in nature.

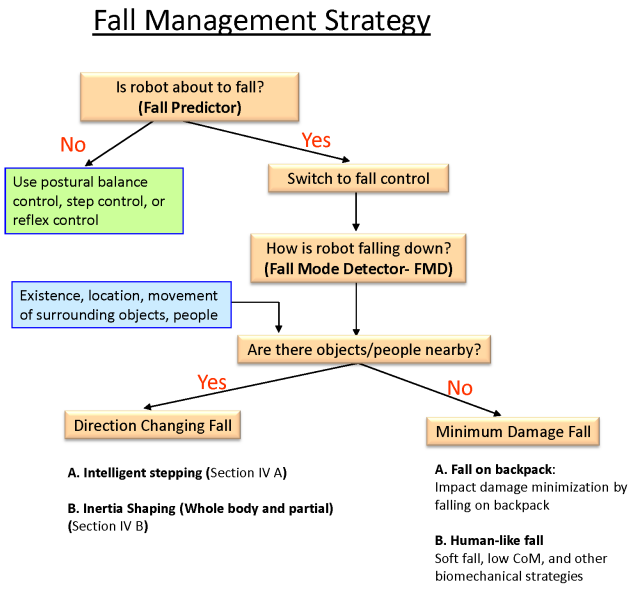


Fig. 2: A schematic diagram showing the essential decision making process of a humanoid fall controller.

3 Overview of Fall Control Strategy

The essential layout of the decision making process of a humanoid fall controller can be represented using a flowchart as shown in Fig. 2. The robot is assumed to contain a fall predictor module which is always alert and is continuously evaluating the robot's state of balance in the form of a simple question: "Is the robot about to fall?" The robot is also assumed to have a standard postural balance controller and possibly a step or a reflex controller for handling stronger disturbances. Most of the time, the fall predictor responds in the negative and the robot performs its planned task using the available balance controllers. When the fall predictor responds in the positive it implies that all strategies to maintain or restore the upright balance of the robot are guaranteed to fail [19] and that the robot faces an inevitable fall. This happens when the robot has exited the so called *fall trigger boundary* (FTB), which separates the balanced state of the robot and the states that lead to a fall.

As soon as the robot state breaches the FTB, the robot must give up trying to restore balance and immediately switch to a fall controller. However, in order to properly select a fall controller the robot needs to analyze additional information. Regardless of the final choice of the fall strategy the robot first analyzes its state of falling using a module we have named the Fall Mode Detector (FMD). This module computes and estimates a number of quantities such as the robot's state of ground contact (single support or double support),

height of its CoM from ground and the CoM velocity, lean angle, direction of fall, the length of time the falling robot takes to touch the ground, etc. [19].

Next, the robot determines if there are objects in its immediate surrounding that are likely to come in contact during the fall. If the surrounding is empty, the robot adopts a minimum damage fall strategy designed to minimize the mechanical damage to its own system. If, however, objects or people are detected in its vicinity, the robot uses the information of their locations and sizes to formulate a direction-changing fall control strategy that can avoid contacting these objects during the fall. This last item is the topic of the current paper.

The direction-changing fall controller uses two basic strategies to change the default fall direction of a robot, while the robot is falling down, as described below.

1. *Foot placement strategy:* This strategy attempts to place the robot feet in an optimal manner in order to change the geometry of the foot support polygon with respect to locations of surrounding objects. An appropriate foot support polygon can minimize the angular deviation between the robot's estimated and desired fall directions. This technique is described in Section 4.1.
2. *Inertia shaping strategy:* Inertia shaping technique reconfigures the robot's overall inertia through joint position control in order to generate angular momentum that nudges the robot towards the desired fall direction. This is described in Section 4.2.

Before we can implement the above-mentioned fall strategies, we need to compute and estimate a number of additional quantities as described below. These are the quantities computed in the FMD module, in addition to sensed information about robot's environment.

1. *Desired fall direction:* Given the location and size of the surrounding objects, the robot computes the most favorable fall direction by assigning a merit score to each available direction. Sections 5.3 and 5.4 describe how this merit score is computed and employed.
2. *Control duration:* The robot estimates the length of time after an inevitable fall is predicted, during which the controller is assumed to remain active. Typically, the control duration is heuristically set as a fraction of the total estimated time the robot would take to fall to the ground. See Section 5.2.1.
3. *Reference point:* The reference point is the location on the ground towards which the robot is estimated to fall at the end of the control duration, based on a reduced-order inverted pendulum model. This estimation makes use of robot's configuration and its current CoM velocity. See Section 5.2.2 for details.

The next section provides the details of the direction-changing fall control strategies.

4 Fall Direction Change through Foot Placement and Inertia Shaping

Our fall direction change controller uses two basic strategies which can be employed either independently, sequentially or simultaneously. The strategies exploit the following two basic observations: first, regardless of its complex motion, a falling humanoid topples predominantly about one of the edges of its support area[1]. Changing the robot's support area geometry can profoundly influence its fall direction. The support area can be modified through the lifting of a foot or through a stepping action, and the specific parameters for these actions are selected using a brute-force search process. This is called the foot placement strategy. The second observation is that a change in the robot's overall inertia can further affect its fall direction through a redirection of its linear and angular momenta. We achieve this by using inertia shaping [24] techniques.

4.1 Support area geometry change through foot placement

Without any fall control, the direction of fall of a humanoid robot is determined by the location and movement of the Center of Pressure (CoP) relative to the support area boundaries. The support area can be approximated by a polygonal area which is the convex hull of all the contact points between the robot feet and the ground. When the robot starts to topple, its CoP touches an edge of the support area called the *leading edge*. Therefore, a change in the physical location of the leading edge of the support area with respect to the robot's CoM exerts influence on the direction of rotation of the robot, *i.e.*, the direction of fall.

In the schematic diagram of Fig. 3, a humanoid robot is shown subjected to a forward push as indicated by the red arrow. If the push is strong enough to topple the robot, the CoP will approach the front edge (red dotted) of the support area and the robot will begin to rotate about this leading edge.

The direction and magnitude of the toppling motion is given by PQ where P is the CoP and Q is what we call a *reference point*. The reference point should indicate the direction of fall. In this paper, we have

[1] The robot can temporarily topple about a vertex of the foot support polygon.

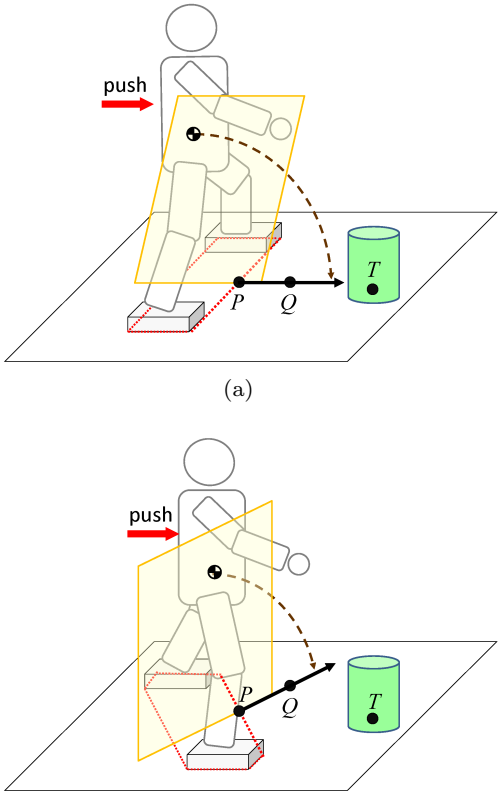


Fig. 3: A schematic diagram showing the basic idea behind direction-changing fall control through support base geometry modification. A forward push on the robot from the back is assumed. P denotes the CoP, and Q is the reference point (capture point, explained in section 4.1). The dotted lines show the support base (polygonal convex hull) of the robot while the polygon edge containing CoP is red dotted.

used the *Capture Point* [28] as the reference point[2]. Although PQ may not be initially perpendicular to the leading edge of the support area, it becomes so once the toppling motion sets in.

With a different geometry of the support base as in Fig. 3(b), for the same push, the robot would rotate about a new leading edge and fall in the new direction PQ. If the robot is to avoid falling on an object in front of it, we can effect a change in the fall direction by changing the support base (specifically, its leading edge) from Fig. 3(a) to Fig. 3(b).

In practice, there are two major challenges in successfully executing this motion. First, the robot becomes underactuated as soon as it starts toppling. This creates 1 or 3 uncontrolled degrees of freedom (DoF) depending on whether the robot is toppling about an edge or a corner. Therefore, we should design a controller very carefully to deal with this underactuated phase. Second, the CoP and the reference point continuously move as the robot moves, which might change the leading edge during fall.

We can make the robot step at a desired location by controlling its leg joint velocities through inverse kinematics. We have the following relationships relating the velocities of the left and right feet and the robot body:

$$\boldsymbol{V}_L - \boldsymbol{V}_{body} = \boldsymbol{J}_L \dot{\boldsymbol{\theta}}_L \tag{1}$$

$$\boldsymbol{V}_R - \boldsymbol{V}_{body} = \boldsymbol{J}_R \dot{\boldsymbol{\theta}}_R, \tag{2}$$

where $\boldsymbol{V}_L$, $\boldsymbol{V}_R$ and $\boldsymbol{V}_{body}$ are (6×1) vectors containing linear and angular velocities of the left and right foot, and of the body frame, respectively. $\dot{\boldsymbol{\theta}}_L$ and $\dot{\boldsymbol{\theta}}_R$ are 6×1 joint velocity vectors of the left and right legs[3], respectively, and $\boldsymbol{J}_L$ and $\boldsymbol{J}_R$ are the leg Jacobian matrices.

Subtracting Eq. 1 from Eq. 2:

$$\boldsymbol{V}_{R-L} = [\boldsymbol{J}_{R-L}] \left[\dot{\boldsymbol{\theta}}_R,\ \dot{\boldsymbol{\theta}}_L\right]^T \tag{3}$$

where $\boldsymbol{V}_{R-L} = \boldsymbol{V}_R - \boldsymbol{V}_L$ and $\boldsymbol{J}_{R-L} = [\boldsymbol{J}_R, \quad -\boldsymbol{J}_L]$ is the (6×12) matrix called the foot-to-foot Jacobian.

The necessary joint velocities $\dot{\boldsymbol{\theta}} = \left[\dot{\boldsymbol{\theta}}_R, \quad \dot{\boldsymbol{\theta}}_L\right]^T$ to move the swing leg to the desired location are given by:

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^{\#}_{R-L}(\boldsymbol{V}_R - \boldsymbol{V}_L) \tag{4}$$

where, $\boldsymbol{J}^{\#}_{R-L}$ is the pseudo-inverse (damped least square [2]) of $\boldsymbol{J}_{R-L}$.

4.1.1 Estimation of the allowable stepping zone

The allowable stepping zone D is the region on the ground anywhere inside which the robot can plant its foot within the control duration time ΔT. As shown in Fig. 4 the allowable stepping zone is approximated as a rectangular area with sides $2D_x$ and $2D_y$. At each point on this rectangle the robot foot can rotate through a range of orientation from $-D_\beta$ to D_β. The values of D_x, D_y and D_β are computed using the following equations:

$$D_x = \Delta T \sum_{i=1}^{12} \left|\boldsymbol{J}_{R-L}(x,i)\dot{\theta}_i^{MAX}\right| \approx \gamma \Delta T \sum_{i=1}^{12} \left|\boldsymbol{J}_{R-L}(x,i)\right|, \tag{5}$$

$$D_y = \Delta T \sum_{i=1}^{12} \left|\boldsymbol{J}_{R-L}(y,i)\dot{\theta}_i^{MAX}\right| \approx \gamma \Delta T \sum_{i=1}^{12} \left|\boldsymbol{J}_{R-L}(y,i)\right|, \tag{6}$$

$$D_\beta = \Delta T \sum_{i=1}^{12} \left|\boldsymbol{J}_{R-L}(\beta,i)\dot{\theta}_i^{MAX}\right| \approx \gamma \Delta T \sum_{i=1}^{12} \left|\boldsymbol{J}_{R-L}(\beta,i)\right|, \tag{7}$$

[2] More detail about Capture Point is included in Section 5.2.2.

[3] We assume 6-DoF legs.

where $\dot{\theta}_i^{MAX}$ is the maximum velocity of the i^{th} leg joint and i corresponds to one of the 12 joints of the two legs. $\boldsymbol{J}_{R-L}(k,i)$ is the numerical value of (k,i) element of Jacobian $\boldsymbol{J}_{R-L}$, which is computed at the time of control trigger. Note that $k \in [x, y, \beta]$ corresponds to the rows 1,2, and 6, respectively of $\boldsymbol{J}_{R-L}$. Finally γ is a constant used to approximate $\dot{\theta}_i^{MAX}$, which is assumed same for all joints.

To compute the best foot placement location we divide the allowable stepping zone into a number of small cells. Each cell corresponds to a foot position given by (x, y) and contains a range of foot orientation angles given by β. The dimension of each cell is selected manually through trial and error. We re-plant the non-support foot according to (x, y, β) of each cell in simulation, and estimate a new reference point and a new CoP. We repeat this step at each cell to find the optimal new CoP.

In practice we use only the upper half of the allowable stepping zone cut by the inclined separatrix line which is perpendicular to PQ and goes through the center of the moving foot, as shown in Fig. 4. This is because when a robot is falling towards PQ it can hardly place its foot on the other side of the separatrix. This area, shaded in yellow in Fig. 4, is divided into cells of foot position (x, y).

The angle of deviation between the desired and the estimated fall direction is computed for each case and the optimal CoP is selected as the one that results in the smallest deviation. We assume a polygonal foot sole and the support polygon can be computed with a finite number of points. The reference point needs to be estimated at the time the non-support foot touches the ground.

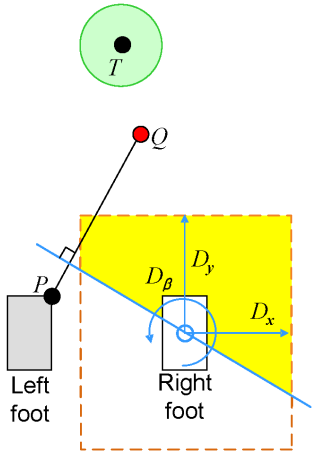


Fig. 4: The allowable stepping zone is shown by the yellow shaded area. The left foot is the support foot. P is the CoP when the robot is in the single support phase and Q is the reference point. T represents an object for the robot to avoid falling on. The allowable stepping zone is the yellow-shaded upper part of the rectangle (above the inclined blue separatrix line) with the maximum half-side lengths D_x and D_y. D_β denotes the maximum amount of rotation of the swing foot.

4.1.2 Step controller for a toppling humanoid

Recall that optimal stepping corresponds to the robot stepping on a location on the ground and with a foot orientation angle that results in the minimum angular deviation between the estimated and the desired fall direction. Note that this optimal solution does not imply optimality in a global sense. Also, the controller does not guarantee the optimal solution, but rather it tries to achieve the best fall angle given the strategies.

Because the available time is very short, the estimation is done using inverted pendulum models. Once the optimal step location is computed, one could hope to simply control the joint angles through inverse kinematics. However, taking a successful step to the optimal step location is not trivial because the standard inverse kinematics solution will not be sufficient for a toppling robot. There are two reasons for this. First, the support foot of the toppling robot is not flat with the ground; therefore, the computation of the position and orientation of the stepping foot based on robot joint angles will be inaccurate. Moreover, the toppling of the robot foot makes the robot underactuated because of the passive joint created at the foot/ground contact. In simple terms, underactuation makes the robot kinematics influenced by its dynamics in non-intuitive ways, and a simple position controller is not likely to succeed in making the robot step as planned.

To deal with this, one solution might be to implement a controller that directly models and controls the robot's state of underactuation [8]. However, such controllers are typically computationally expensive, a luxury we do not have for the current application. What we currently do is to continuously estimate the rotation angle of the robot's stance foot and add appropriate correction in the control of its stepping foot. Assuming that the robot possesses an IMU in the trunk, the foot rotation angle can be estimated by noting the mismatch between the trunk orientation angles as computed by the IMU and by the robot joint angle sensors. With this information, we implement a leg controller to ensure that the swing foot is flat as it touches down on the ground.

Since we assume that the CoP does not change during the fall, the CoP is modeled as a passive rotational joint about which the support foot rotates, as shown in Fig. 5. The support foot rotation is estimated using the inverted pendulum models *without* control. The trans-

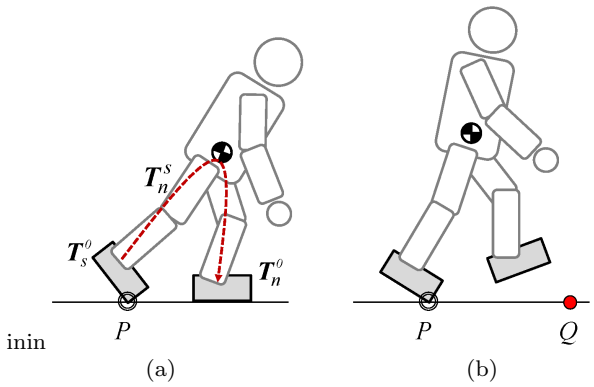


Fig. 5: (a) Desired *future* landing posture after the fall control time. $\boldsymbol{T}_s^0$ and $\boldsymbol{T}_n^0$ are the transformation of the support and the non-support foot with respect to the global frame. $\boldsymbol{T}_n^s$ is the desired transformation matrix from the support foot to the non-support foot to attain the desired landing posture. (b) Desired *current* robot posture to realize the desired landing posture. For a robot that is toppling, the support foot rotates about the CoP, P, towards the reference point Q. We model a free joint at P. Without external forces, the joint angle should increase. The leg joints are controlled to satisfy $\boldsymbol{T}_n^s$ computed in (a).

formation of the non-support foot with respect to the global frame $\boldsymbol{T}_n^0$ is determined by the desired stepping location. The transformation of the support foot $\boldsymbol{T}_s^0$ is estimated by simulating the inverted pendulum model. Therefore, the desired transformation matrix from the support foot to the non-support foot $\boldsymbol{T}_n^s$ can be computed as,

$$\boldsymbol{T}_n^s = (\boldsymbol{T}_s^0)^{-1}\boldsymbol{T}_n^0. \tag{8}$$

If the joints are controlled according to $\boldsymbol{T}_n^s$ before the non-support foot hits the ground as in Fig. 5(b), the robot is expected to step on the desired location by gravity, Fig. 5(a). We can compute joint velocities needed to move the swing leg using Eq. 4.

Although an appropriate stepping action (including foot lifting) can exert powerful influence on the future motion of the falling robot, in some cases it still may not be sufficient to control the robot to fall in the right direction. In these cases we employ a approach called inertia shaping, which attempts to systematically control the robot's centroidal angular momentum through the kinematic control of its overall inertia matrix. We describe inertia shaping in the next section.

4.2 Inertia shaping

The humanoid can attempt to further change the fall direction after a step is taken. Since a falling robot is normally underactuated, direct control of the CoM would not be effective in general. However, we can indirectly change the fall direction by generating angular momentum. For this, we have developed a technique called inertia shaping [24].

In inertia shaping, we control the centroidal composite rigid body (CRB) inertia [38] or the locked-inertia of the robot. Centroidal CRB inertia is the instantaneous rotational inertia of the robot, referenced at its CoM, if all its joints are locked. Unlike linear inertia, which is always constant, the CRB inertia is a function of the robot configuration and can vary continuously.

Approximating the robot as a reaction mass pendulum, RMP [24], or an inverted pendulum with rigid body inertial mass, and assuming no slip at the ground, its CoM velocity $\boldsymbol{V}_G$ can be computed as (see Fig. 6):

$$\boldsymbol{V}_G = \omega_G^P \times \boldsymbol{PG} \tag{9}$$

where G and ω_G^P are the CoM location and the angular velocity of the inverted pendulum, respectively. For best results, we want $\boldsymbol{V}_G = -c\,\mathbf{PT}$ for some scalar c. [4] This can be achieved by setting the desired angular velocity ω_d as follows,

$$\omega_d = -k\,(\boldsymbol{e}_{\boldsymbol{z}\times\boldsymbol{PT}})\,, \tag{10}$$

where $\mathbf{e}_{\boldsymbol{z}\times\boldsymbol{PT}}$ is a unit vector along the cross product of $\boldsymbol{z}$ and $\boldsymbol{PT}$, and k is the magnitude of angular velocity. The desired locked inertia is obtained as $\boldsymbol{I}_d = \boldsymbol{R}\boldsymbol{I}\boldsymbol{R}^{-1}$, where $\boldsymbol{I}$ is the current locked inertia and $\boldsymbol{R}$ is the rotation matrix obtained with an exponential map [25] from ω_d:

$$\boldsymbol{R} = \exp(\Omega_d), \tag{11}$$

where Ω_d is the skew-symmetric matrix corresponding to ω_d.

To implement inertia shaping, we string out the 6 unique elements of the CRB inertia matrix in the form of a vector: $\boldsymbol{I}_{(3\times3)} \rightarrow {}^s\widehat{\boldsymbol{I}}_{(6\times1)}$. Next, we obtain the *CRB inertia Jacobian* $\boldsymbol{J}_{\boldsymbol{I}}$ which maps changes in the robot joint angles into corresponding changes in ${}^s\widehat{\boldsymbol{I}}$, *i.e.*,

$$\delta\,{}^s\widehat{\boldsymbol{I}} = \boldsymbol{J}_{\boldsymbol{I}}\delta\boldsymbol{\theta}. \tag{12}$$

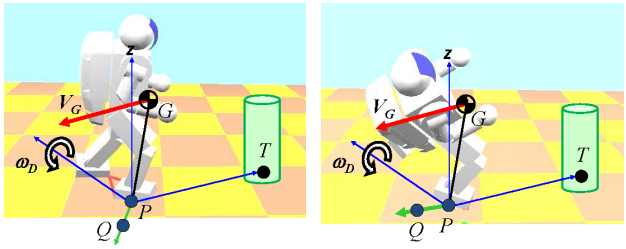
To attain $\boldsymbol{I}_d$, the desired joint velocities are given by:

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}_I^{\#}(\boldsymbol{I}_d - \boldsymbol{I}) \tag{13}$$

where $\boldsymbol{J}_I^{\#}$ is the pseudo-inverse of $\boldsymbol{J}_I$.

The humanoid can recruit all the joints to attain $\boldsymbol{I}_d$. The effect of inertia shaping might not always be big

[4] Extension to a general case with multiple objects such as in Fig 16 is trivial once the desired fall direction is chosen.



(a) (b)

Fig. 6: a) To avoid falling on the cylindrical object located at T, the desired linear velocity of the robot's CoM, $\boldsymbol{V}_G$, should be towards $\overrightarrow{TP}$, where P is the CoP. b) To achieve this the robot should overall rotate about an axis obtained by the cross product of PT and the vertical.

enough to obtain the desired CoM velocity $\boldsymbol{V}_G$, however, even a modest change is sometimes very useful.

Equation 13 is used for whole body inertia shaping, which cannot be launched before stepping (foot placement) is completed, because the two actions may be in conflict. This can sometimes lead to the the loss of useful time during which the robot's upper body does not contribute to attaining a desired fall direction. To correct this situation, we introduce *partial inertia shaping*, which is a procedure to change the CRB inertia of the robot *simultaneously* during foot placement, without using the joints that are involved in the latter.

During partial inertia shaping, we basically recruit only the upper body joints. The CRB inertia Jacobian $\boldsymbol{J}_I$ introduced above can be re-written as:

$$\boldsymbol{J}_I = [\boldsymbol{J}_{PIS},\ \ \boldsymbol{J}_{FP}] \tag{14}$$

where, $\boldsymbol{J}_{PIS}$ is the CRB inertia Jacobian corresponding to the joints that are free from foot placement strategy execution, whereas $\boldsymbol{J}_{FP}$ is the CRB inertia Jacobian corresponding to the joints involved in the foot placement strategy execution. The desired angular velocities $\dot{\boldsymbol{\theta}}_{PIS}$ to attain $\boldsymbol{I}_d$ by partial inertia shaping are given by:

$$\dot{\boldsymbol{\theta}}_{PIS} = \boldsymbol{J}^{\#}_{PIS}\,(\boldsymbol{I}_d - \boldsymbol{I} - \boldsymbol{J}_{FP}\,\dot{\boldsymbol{\theta}}_{FP}) \tag{15}$$

where $\boldsymbol{J}^{\#}_{PIS}$ is the pseudo-inverse of $\boldsymbol{J}_{PIS}$ and $\dot{\boldsymbol{\theta}}_{FP}$ is given by the controller for the optimal foot placement strategy.

Just as the CRB inertia matrix can be defined with respect to any appropriate point such as the CoM or the CoP, the inertia shaping can be performed interchangeably about these respective points. We have earlier performed inertia shaping about the CoM [43]. However, since the desired angular velocity used to derive the desired inertia matrix is computed about the CoP, it is preferable to perform inertia shaping about CoP as well. Moreover, partial inertia shaping about CoP is more effective than that about CoM because the arm and the upper body configurations make more significant contributions to the CRB inertia about CoP. So, the desired inertia matrix $\boldsymbol{I}_d$ derived here is about CoP *i.e.*, $\boldsymbol{I}^P_d$ as shown in Fig. 7.

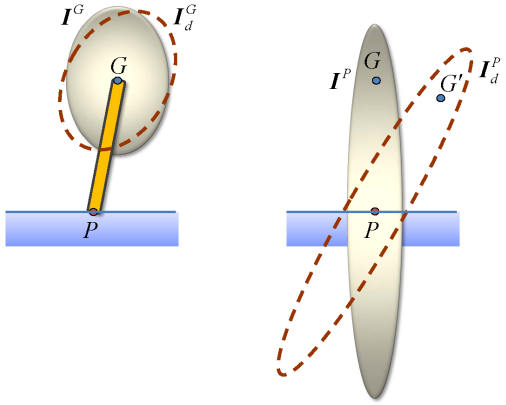


Fig. 7: Comparing inertia shaping about CoM (left) and CoP (right). Shaded solid ellipsoids and ellipsoids with dashed outlines, in each case, denotes current and desired inertia matrices, respectively. Note that inertia shaping about the CoP allows a movement of the CoM (G to G'), which the other does not.

5 Planning and Selection of the Optimal Fall Control Strategy

In the previous section, we described the general nature of the two strategies that we use to change the fall direction of humanoid robots. However, there are additional details to pay attention to: a specific control needs to be selected along with the associated parameters and it is to be executed at a specific time and for a specific duration. Moreover, the implemented control may contain the two strategies executed either independently, sequentially or simultaneously We need a plan to coordinate and supervise these strategies. This section describes this planning process, which is based upon the geometric set-up of the robot in its environment, and the computation of a few supporting quantities mentioned in Section 3.

We have made the following assumptions in formulating and selecting our controller:

– All motors stay active during the entire motion of the robot;

– The robot can perfectly sense its states, including its global position and all the joint angles;
– The robot knows the location and overall physical dimensions of all the objects in its vicinity all the time [5];
– The floor has enough friction to ensure no slip; and
– The robot's feet are polygonal.

The last assumption is required to compute the estimated fall angle for which we use the CoP location. For feet with curved boundaries the CoP location may be continuously fluctuating which may result in unstable computation of fall direction.

Under these assumptions, we discuss the process of prediction of when and how the humanoid would fall, and talk about the selection of proper direction-changing fall controllers according to the falling behavior and the positions and sizes of the surrounding objects.

5.1 Prediction of humanoid fall: Fall trigger boundary (FTB)

The first step in entering a fall control mode is through the fall predictor described in Section 3 and in Fig. 2. The prediction of fall is a critical component of the fall management strategy. The fall predictor helps decide when to switch from a fall avoidance (or balance maintenance) controller to a fall controller. A fall predictor continuously monitors the robot's state, and raises a flag as soon as it predicts an *imminent* fall. A trigger from the fall predictor prompts the robot to abandon the balance maintenance mode, which was just predicted to fail, and to execute a fall control strategy.

As schematically shown in Fig. 8, the FTB of a robot encloses a region in its feature space in which a given balance controller is able to stabilize the robot. *An exit through the FTB is an indication of an unavoidable fall* and this event can be used to activate the switch from the robot's balance controller to a fall controller.

The parameters that characterize the feature space can include both sensor data such as joint angle and ground reaction force (GRF), and any number of computed variables such as CoM and CoP positions, robot lean angle, angular momentum, etc.

5.2 Fall mode detector (FMD)

In order to choose the correct fall strategy, it helps for the falling robot to quickly estimate *how* it is going to fall. For this, we approximate the robot as an equivalent inverted pendulum as shown in Fig. 9. The pendulum connects the CoP and CoM of the robot and has a point mass equal to the robot mass. If the CoP is located on an edge of the support area, the pendulum is constrained to rotate on a plane perpendicular to the edge. In this case, we model the robot as a 2D inverted pendulum. If instead, the CoP is located at a corner, the estimation uses a 3D spherical inverted pendulum model. The 2D pendulum model has a closed-form solution. However, since the 3D pendulum does not have closed-form solutions, we simply simulate its dynamic equations for the period of control duration. Because the control duration is typically very short, this simulation can be adequately handled.

It might appear at first that the 3D inverted pendulum, being a higher dimensional entity, would also exhibit the behavior of the 2D pendulum, and that only the 3D model would be sufficient to model a falling

[5] The actual determination of this beyond the scope of this paper.

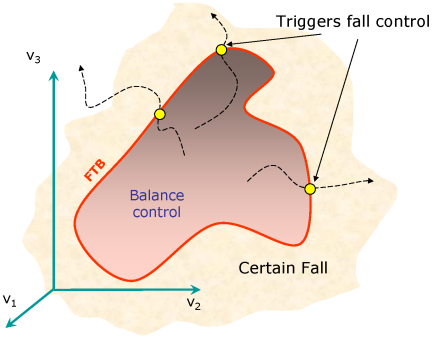


Fig. 8: Schematic of *Fall Trigger Boundary (FTB)*, a boundary in a humanoid feature space that surrounds the region where the humanoid is able to maintain balance. The axes in the figure represent different robot features such as CoM coordinates, angular momentum components, etc. The FTB represents the limit beyond which the robot controller must switch to a fall controller. The shape and size of the FTB are characteristics of a given balance controller.

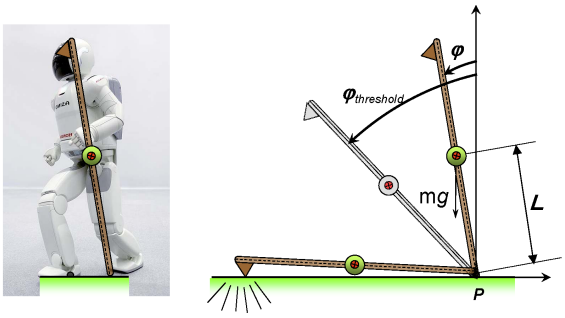


Fig. 9: Simple model of an inverted pendulum falling under gravity. P is CoP, m is the humanoid mass, and ϕ is the lean angle between the CoP-CoM line and the vertical. We use this model for the fast estimation of time duration and other parameters of the robot.

humanoid. However, the motion of a 2D pendulum is constrained in its plane, and it correctly models the motion of a falling humanoid that is toppling about one of the edges of the feet. On the other hand, the motion of the robot, which is toppling about one of the foot vertices, is correctly modeled using a 3D pendulum.

5.2.1 Estimation of control duration

Time-to-fall is a critical parameter for the evaluation and formulation of a fall response strategy. The biomechanics literature contains some data on the time-to-fall of human subjects. A simple forward fall of an adult starting from a stationary 15° inclination takes about 0.98 s, whereas that for a backward fall starting from stationary 5° inclination takes 0.749 s (with flexed knees) and 0.873 s (with extended knees) [37].

The fall controller remains active until its lean angle crosses a certain threshold $\varphi_{threshold}$. We assume that all external forces have disappeared when the robot starts to use the fall controller. The control duration ΔT is obtained through an incomplete elliptic integral of the first kind for the 2D inverted pendulum model [35] when the lean angle goes over the threshold, *i.e.*, ΔT is the time for the pendulum to hit the threshold (not the ground). For the 3D spherical pendulum model, we simulate its dynamic equations to obtain ΔT.

5.2.2 Estimation of reference point

As mentioned before, the capture point is used as the reference point in this work. In order to estimate the capture point at the time ΔT, we need a planar velocity of the robot. Since we use a 2D pendulum model, the planar velocity can be directly calculated from the angular velocity of the pendulum.

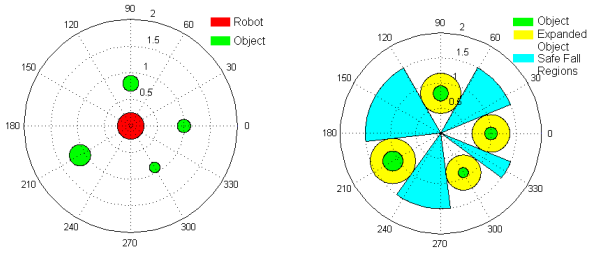
For the 2D linear inverted pendulum model, the velocity after time ΔT is computed from the pendulum energy equation as follows:

$$\dot{\varphi}(\Delta T) = \sqrt{\frac{2E}{I} - \frac{2mgL\cos(\varphi(\Delta T))}{I}} \tag{16}$$

where E is the total energy (constant) of the pendulum, I is the moment of inertia of the pendulum with respect to CoP and L is the distance between its CoP and CoM. For the spherical pendulum, the simulation of the dynamic equations yields the velocity.

5.3 Geometric setup

In 3D space, both the robot and the surrounding objects are approximated by circumscribing vertical cylinders centered at their respective CoMs. On the horizontal projection, the objects are represented by circles and



(a) (b)

Fig. 10: The 2D projection of a humanoid robot surrounded by obstacles of different sizes. (a) The central red circle represents the robot. Its center is located at the robot's CoM and its diameter is equal to robot's maximum leg spread. The circles shown in green are the circumscribing circles of the objects' 2D projections. (b) Following configuration space approach, these latter circles are grown by robot's radius and the robot is reduced to a point. Safe fall regions (cyan cones enclosed by the solid black curves) are the free cones in which the robot's CoM can fall without hitting an object.

the robot is represented by a circle with its center at the CoM and the maximum leg spread as its diameter as shown in Fig. 10(a). We assume that the position and size of the objects are known to the robot at all times. Following the configuration space formulation used in traditional motion planning algorithms [23], the object circles are grown by the radius of the robot circle and the robot is reduced to a point (Fig. 10()b).

The entire planning process uses information in polar coordinates (r, θ) with the point robot at the origin $(0, 0)$, where $r \in \mathbb{R}^+$ represents the distance from the point robot and $\theta \in \Theta = [0, 2\pi]$ represents the direction. The direction $\theta = 0$ represents the reference direction with respect to which all objects' positions and orientations are known. Only objects within a radius of 1.5 times the height of the robot are considered for the planning process and the other objects are considered too far from the robot to be hit.

In this work, the fall direction $\theta_f \in \Theta$ is defined as the vector connecting the robot's initial and final CoM ground projections. The initial state is at control trigger and the final state is the touchdown of the robot with the ground, estimated using inverted pendulum simulations. At fall trigger, all controllers on the robot are assumed to be stopped and the joints are locked with the robot behaving like a rigid body until control trigger is reached. After control trigger is reached, the only active controller is the safe fall controller. The fall direction is independent of the intermediate configurations of the robot, which implies that it is independent of the CoM position during fall.

A safe fall region, characterized by an object-free cone, is the set of continuous fall directions containing

no objects inside them as depicted by the cyan cones enclosed by the solid black lines in Fig. 10(b). These represent the set of directions in which the robot can fall without hitting an object.

5.4 Selection of the optimal fall control strategy

Each fall direction θ_{f_i} receives two scores (s_1^i, s_2^i), whose weighted sum gives the total score $s^i = w_1 s_1^i + w_2 s_2^i$, where $w_1 + w_2 = 1$. Intuitively s_1 represents how much relatively safe the fall direction would be, and s_2 shows how close the fall direction would be to the middle of the safe cone.

Note that the total score s^i is zero when the fall direction θ_{f_i} is at the bisector of the largest safe fall region. Therefore, lower the score, safer is the fall direction. w_1 and w_2 are positive weights, and the equal weights ($w_1 = w_2 = \frac{1}{2}$) are used in the simulations of this paper.

The planner evaluates and selects from three foot placement strategies: a) No Action, b) Lift a Leg and c) Take a Step.

- **No Action:** There is no attempt at controlling the robot beyond locking all joints and letting the robot fall down as a rigid body. This strategy is adopted when the default fall direction of the robot is already deemed safe.
- **Lift a Leg:** This strategy is evaluated only when the robot is in double-support phase. It involves two mutually exclusive sub-strategies, 1) Lift the left leg and 2) Lift the right leg. Lifting a leg reduces the extent of support base to a single footprint. Although simple, this strategy can exert significant influence on the toppling motion.
- **Take a Step:** This strategy involves taking a step from the robot's current position. The number of possible stepping locations provides a number of sub-strategies to be evaluated. The control duration ΔT calculated earlier in the paper is used to estimate the allowable stepping region.

Inertia shaping strategies, presented in Sec. 4.2, are sometimes used in conjunction with, and at other times as a replacement for, the foot placement strategies. We classify these strategies as follows:

- **Whole Body Inertia Shaping:** This strategy recruits all robot joints and employs inertia shaping on the entire robot. The inertia shaping strategy replaces the foot placement strategy when it fails to produce a safe fall.
- **Partial Inertia Shaping:** This strategy employs inertia shaping using only those joints that are not involved in the stepping.

5.5 Strategy selection flowchart

The strategy selection is done as presented in the flowchart in Fig. 11. In case of steady fall, the fall direction estimation is more accurate and the *No Action* and *Lift a Leg* strategies are given preference over the *Take a Step* strategies because the former are guaranteed for a successful completion. In case of unsteady fall or when the *No Action* and *Lift a Leg* strategies fail to produce safe fall, all foot placement strategies are evaluated and their estimated fall directions are assigned scores. The strategy with the minimum total score is chosen to be the optimal safe fall direction. As one can see, even when no foot placement strategy produces a safe fall direction, the algorithm chooses the strategy with the lowest score that corresponds to the fall direction closest to the safe fall region.

When no foot placement strategy produces safe fall, partial inertia shaping strategy is coupled with the optimal foot placement strategy. The bisector of the safe fall region closest to the direction corresponding to the optimal foot placement strategy is chosen to be the desired direction for the partial inertia shaping procedure. This fall direction corresponds to the local minima closest to the current fall direction. While the optimal foot placement strategy tries to do the best it can, the partial inertia shaping procedure tries to move the body to the closest safe fall region.

The strategy selection procedure described above happens only at control trigger whereas the strategy execution happens after it. At any future time after the execution of the chosen strategy, if the robot's fall direction is still unsafe, the whole body inertia shaping is initiated, because that is the only method available at that point. The bisector of the safe fall region closest to the current fall direction is chosen to be the desired direction of fall and the inertia shaping procedure tries to achieve it.

Finally, if the robot's lean angle exceeds a maximum threshold, all the motors are turned off, *i.e.*, all joints are unlocked, in order to reduce the damage to the motors due to impact.

6 Simulation Results

Our fall controller has been executed in the Webots simulation environment for a human-sized humanoid and an Aldebaran NAO robot. In order to handle the different platforms seamlessly in simulation and experiment, we developed a modular software architecture which focuses on an interface to connect a robot controller to specific applications.

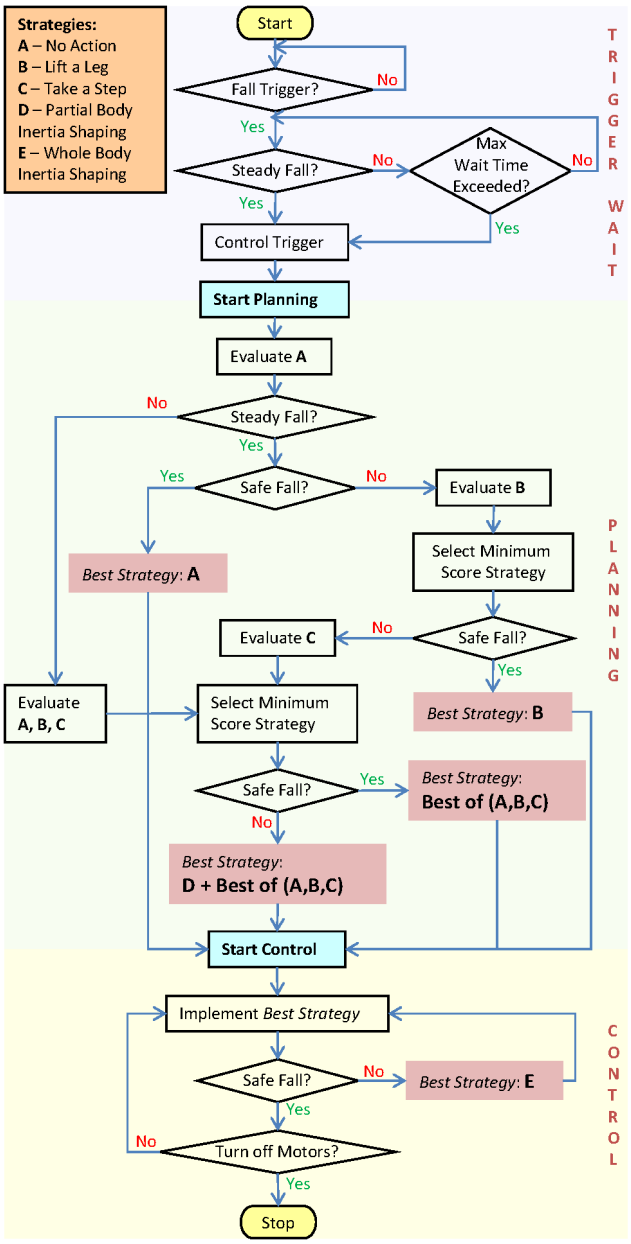


Fig. 11: Decision making procedure for safe fall planning and control.

6.1 Full-size humanoid robot

The full-sized humanoid robot that we used for simulation study is 1.3 m tall and weighs approximately 54 kg. The robot possesses 26 DoF and each leg has 6 DoF. The resolutions of the cells used in Eq. 7 are 50mm for (x, y) and 0.1 rad for β. A control sampling time is 1 msec. We assume that the global locations and orientations of the robot bodies are given.

6.1.1 Fall while avoiding a single object

We start from direction-changing fall with a single object, since a single object means a sole desired fall direction that is obviously opposite to the object direction. Unlike a cluttered environment, in this case we can easily see the desired direction and compare it with results from our fall controller.

Figure 12 shows snapshots of the simulation, where the humanoid stands on both feet and is subjected to a push on its trunk for 0.1 s. The push has a magnitude of 200 N forward and 50 N to the right. The target for the humanoid to avoid is located 1.2 m in front of it, and the head of the humanoid would hit it without any fall control. The fall controller starts 0.05 s after the push has ended. Inertia shaping, if used, begins to work as soon as the swinging foot contacts the ground. During direction-changing fall control, the support base changes from a rectangle to a point, then to a quadrilateral and back to a rectangle, as shown in the bottom row of Fig. 12.

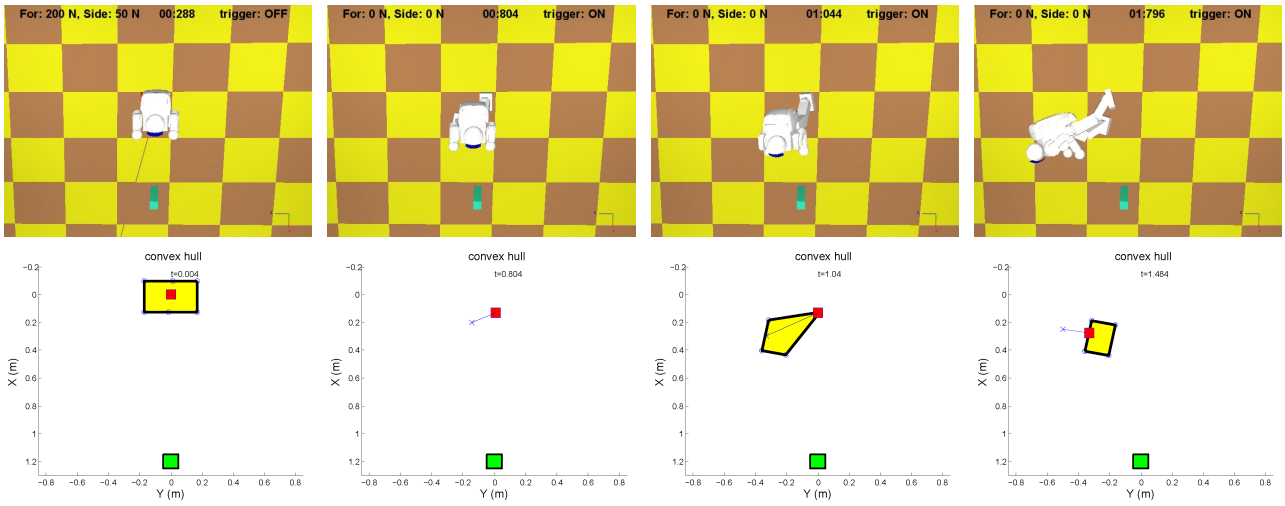
The direction of fall changes, as expected, according to support base geometry change. When the humanoid is on double support, it topples forward and rotates about the front edge of the support base for which the CoP is located roughly in the middle. Once the robot lifts the right leg to take a step, it starts toppling around the right top corner of the left foot and the support base shrinks to a point, as shown in Fig. 12(b). Taking a step makes the support base polygon a quadrilateral, as shown in Fig. 12(c), and the direction of fall goes to the right since the reference point is at the right of the support polygon. Finally, the humanoid achieves the rightward fall direction.

Figure 13 shows the motion of a falling humanoid which employs both the support base geometry controller and whole body inertia shaping controller. After taking a step as shown in Fig. 13(b), the humanoid changes the fall direction by rolling the upper body backward, see Fig. 13(c).

Comparison of the CoM trajectories for the three different cases of no control, support polygon change, and support polygon change plus inertia shaping are shown in Fig. 14. The figure clearly shows that the trajectory of the full controller diverges from the trajectory of the support polygon change and goes backwards.

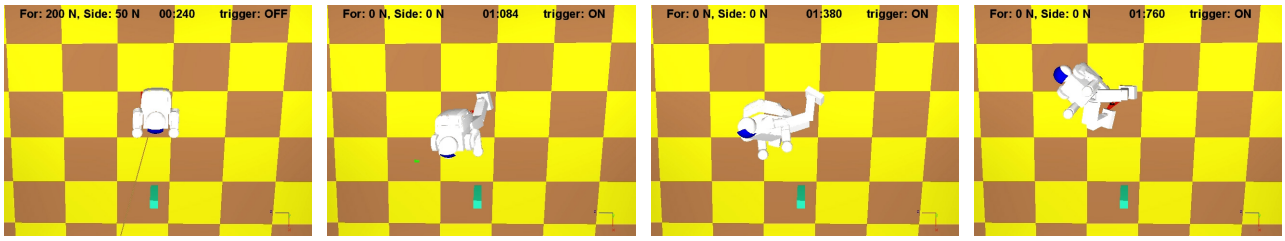
6.1.2 Fall while avoiding multiple objects

In this case, the robot's environment contains four objects as shown in Fig. 15. The robot is pushed at the CoM of its trunk with horizontal forces of different magnitudes and directions; and the performance of the safe



(a) (b) (c) (d)

Fig. 12: Top row pictures are snapshots of falling humanoid with changing support polygon. The bottom row figures show the changing support polygon, the CoP and the capture point. The yellow region is the support polygon. The green square is the object to avoid. The small red square is the CoP, and the cross mark is the reference point. These two points are connected by the blue line which also shows the estimated direction of fall. (a) The humanoid gets a forward push. The direction of the push is shown by the red arrow from the body. The support polygon is a rectangle formed by the two feet. (b) The humanoid has lifted the right foot to take a step. Since the robot is toppling its support polygon is simply a point, and it is coincident with the CoP. The reference point implies that the falling direction is toward left. (c) The humanoid has taken a step. The support polygon is a quadrilateral formed by four points of the right foot and the right-forward corner of the left foot. (d) The humanoid is falling towards its right, rotating about the rightmost edge of the right foot.



(a) (b) (c) (d)

Fig. 13: Snapshots of a falling humanoid which uses both step controller and whole body inertia shaping controller. The push is the same as in Fig. 12. Inertia shaping starts after taking a step. The humanoid appears to lean its body backwards as if it does limbo. After the conclusion of inertia shaping, the humanoid has fallen almost backwards.

fall controller for each case was analyzed. All forces are exerted for a duration of 100 ms.

When the robot is pushed with a backward force of 210 N, the default fall is already safe. Our planning procedure successfully detects steady fall and chooses *No Action* as the best strategy, which results in a safe fall as shown in Fig. 15. Figure 16 shows the safe fall behavior as a result of choosing *Lift a Leg* strategy after identifying a steady fall when pushed with a forward force of 210 N.

Figure 17 shows the safe fall behavior as a result of choosing *Take a Step* and *Partial Inertia Shaping*. As expected, we can see significant arm motions in this case, and the robot falls in the forward left direction.

In order to emphasize the significance of *Partial Inertia Shaping*, we compare the outcomes of different strategy executions with respect to identical pushes. In Fig. 19, we compare *No Action*, *Take a Step* and *Partial Inertia Shaping* strategies when the robot was pushed with a forward force of 235 N, for a duration of 100 ms. The CoM trajectories show that the safe fall was produced by using *Partial Inertia Shaping* coupled with *Take a Step* strategy.

All the above results are for cases where the robot was standing stationary upright when pushed. We also successfully tested the fall control strategy for a few cases where the robot was pushed *during walking*. One result is shown in Fig. 20. The main objective of this exercise was to insure that the control algorithm works

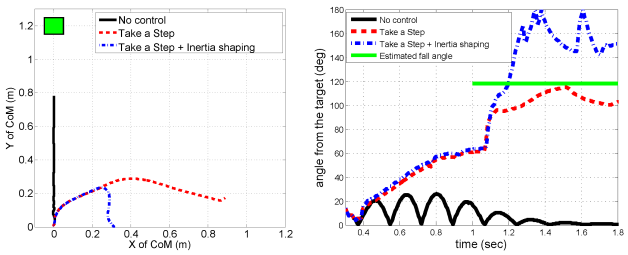


Fig. 14: Simulation plots of CoM trajectories (Left) and avoidance angles (Right) of a falling humanoid which was pushed during upright standing. The avoidance angle is for the line from the CoP to the CoM. The humanoid falls on the single target without any control, which corresponds to a 0° avoidance angle. Intelligent stepping improves this to 100° and inertia shaping further improves to 180°. The avoidance angle between the robot's fall direction and the direction of the nearest obstacle is computed using the lean line which extends from the CoP to the CoM. The oscillation of the avoidance angle is the result of the oscillation of the CoP, which is often caused by a rocking motion of the robot when it does not have a firm and stable contact with the ground.

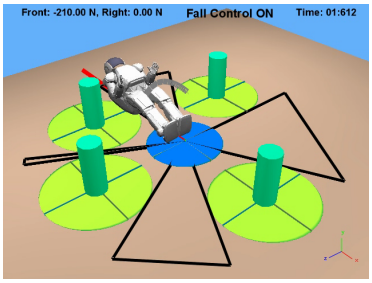
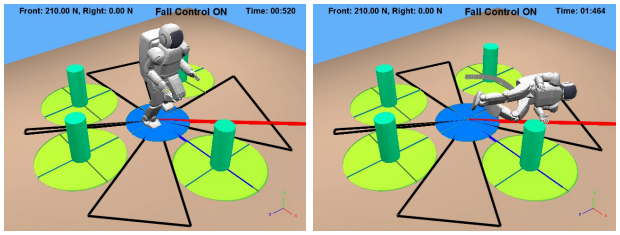
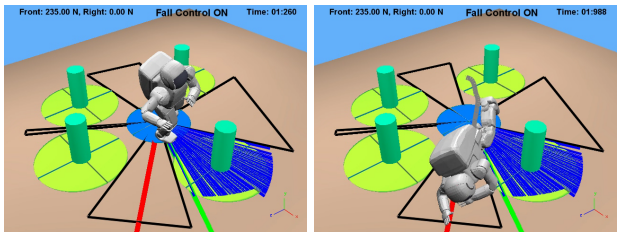


Fig. 15: The robot is pushed with a backward force of 210 N, for a duration of 100 ms. Default fall direction is already safe. No action is taken to change the fall direction.



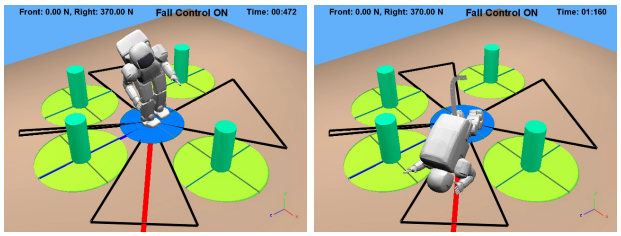
(a) (b)

Fig. 16: The robot is pushed with a forward force of 210 N, for a duration of 100 ms, at the CoM of its trunk. The default fall direction is towards the object in the front. a) The robot lifts the left leg to change fall direction; b) this results in a safe fall at an empty space.



(a) (b)

Fig. 17: Robot is pushed by a forward force of 235 N for 100ms. (a) The robot is taking a step and moving arms to perform *Partial Inertia Shaping*. (b) Safe fall as a result of *Take a Step* and *Partial Inertia Shaping* Strategies. The green line shows the unsafe fall direction if partial inertia shaping was not used.



(a) (b)

Fig. 18: The robot is pushed with a force of 370 N for a duration of 100 ms, to its right. (a) *Whole Body Inertia Shaping* starts when no foot placement strategy produces safe fall and (b) Safe fall as a result.

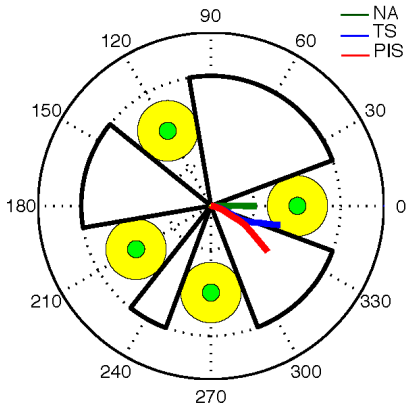
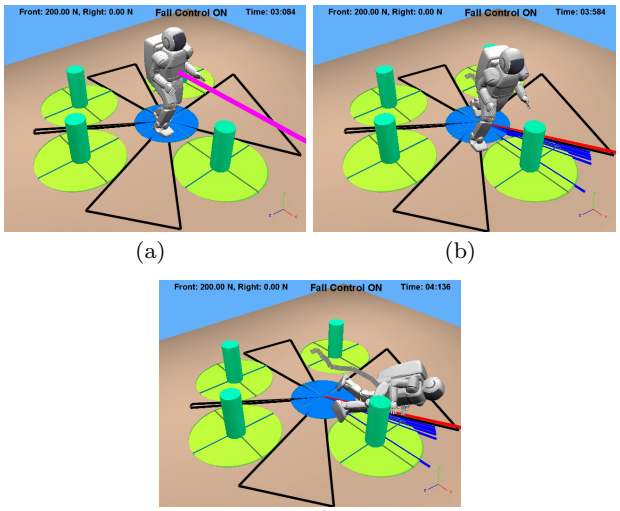


Fig. 19: Comparing the performances of different strategies. *NA - No Action*, *TS - Take a Step*, and *PIS - Take a Step + Partial Inertia Shaping*. The CoM trajectory during each strategy execution is shown. The robot is pushed with a forward force of 235 N. Only PIS produces safe fall.



(a) (b) (c)

Fig. 20: (a) The robot is pushed with a 200 N forward force while walking, (b) Take a Step was the optimal strategy. This stepping location is different from the step the robot was about to take while walking. (c) Resulting safe fall.

in the walking regime. However, we simulated a very slow gait in which the robot took a 15cm long step in 1.5sec, resulting in a 0.1m/sec speed for the feet and 0.75m/sec speed for the body. In this case, because the robot is already in the single support state, it cannot use the *Lift a Leg* strategy. Other than this difference, the rest of the fall strategy described in Fig. 11 was employed unchanged and without any special modification for non-stationary conditions. More exhaustive experiments of humanoid fall during gait is necessary and is planned for the future.

Note that the location and direction of the push on the humanoid body will change the optimal solution of the direction-changing fall because they will affect the states of the reduced model, the 2D and 3D inverted pendulums, which are used in the fall controller. For example, the same push to the head of the robot will result in higher rotational velocity compared to a push at a lower point on the body, Consequently, the results can be quite different.

6.2 Aldebaran NAO robot

In order to implement the direction-changing control on the NAO robot, we first need to address a few issues specific to this humanoid robot. The NAO robot is 57.3cm tall, possesses 24 DoF and weighs 5.2kg. Each arm of NAO has 5 DoF and each leg has 5 DoF. The head has 2 DoF and the pelvis has a 1 DoF joint. Also, each hand has a 1 DoF joint.

The kinematic structure of this robot is unique in that its two legs have the shared joint which connects the body to both legs. The actuator of the shared joint is located at the inside of the hip and rotates the two first joints of the both legs at the same rate, therefore the two joints cannot be controlled independently. Consequently, our inverse kinematics and inertia shaping algorithms must be updated.

One way to treat this shared joint is to imagine an asymmetry in the legs where one leg has 6 DoF and fully possesses the pelvic joint while the other leg has 5 DoF. This asymmetry raises a problem in solving the inverse kinematics since most humanoids have two 6 DoF legs and the typical inverse kinematics solution for a 6 DoF link can be used for both legs. In order to use our fall controller on NAO, we design a Jacobian-based inverse kinematics for this special joint configuration, which enables stepping as well as control of the body posture.

Suppose that both legs of the robot have firm support on the ground. In Eqs. 1 and 2, $\boldsymbol{\theta}_L$ is a 5×1 joint angle vector of the left leg, and $\boldsymbol{\theta}_R$ is a 6×1 joint angle vector of the right leg. Consequently, the foot-to-foot Jacobian matrix $\boldsymbol{J}_{R-L}$ becomes a 6×11 matrix. Note that any one of the legs could be considered 6 DoF while the other is 5 DoF.

In order to simultaneously control the location of the body frame (6 DoF) and the step location (6 DoF) we need a total of 12 DoF. This is relatively straightforward when each leg possesses 6 DoF. However, for the NAO H25 robot, we lack 1 DoF because the two legs have a total of 11 DoF. To deal with this we design a cost function for the inverse kinematics algorithm to minimize:

$$\begin{aligned} min \quad & \|\boldsymbol{V}_{R-L} - \boldsymbol{J}_{R-L}\Delta\boldsymbol{\theta}\|^2 + \lambda^2 \|\Delta\boldsymbol{\theta}\|^2 + \\ & \epsilon^2 \|\boldsymbol{V}_{body} - \boldsymbol{J}_R\Delta\boldsymbol{\theta}_R\|^2 , \end{aligned} \tag{17}$$

where $\boldsymbol{V}_{R-L} = \boldsymbol{V}_R - \boldsymbol{V}_L$ in which $\boldsymbol{V}_R$ and $\boldsymbol{V}_L$ are the velocities of the right and left foot, respectively, $\boldsymbol{V}_{body}$ is the velocity of the trunk. These quantities were introduced in Eqs. 1, 2 and 3. λ and ϵ are constant weights. The parameter ϵ controls the relative importance between the step displacement and the body displacement. For example, a low ϵ puts higher priority on the step displacement. λ regulates the control output $\Delta\boldsymbol{\theta}$. Note that this inverse kinematics is another version of the damped least-squares solution [2].

This cost function pursues the simultaneous control of the body location and the step displacement while minimizing the total joint angle displacements. Equa-

tion 17 can be re-written as follows:

$$min \left\| \begin{bmatrix} \boldsymbol{J}_{R-L} \\ \lambda \boldsymbol{I} \\ \epsilon \begin{bmatrix} \boldsymbol{J}_R & 0 \end{bmatrix} \end{bmatrix} \Delta\boldsymbol{\theta} - \begin{bmatrix} \boldsymbol{V}_{R-L} \\ 0 \\ \epsilon \boldsymbol{V}_{body} \end{bmatrix} \right\|^2, \tag{18}$$

which leads to the following inverse kinematics solution:

$$\Delta\boldsymbol{\theta} = \left(\overline{\boldsymbol{J}}^T \overline{\boldsymbol{J}} + \lambda^2 \boldsymbol{I} \right)^{-1} \overline{\boldsymbol{J}}^T \begin{bmatrix} \boldsymbol{V}_{R-L} \\ \boldsymbol{V}_{body} \end{bmatrix}, \tag{19}$$

where

$$\overline{\boldsymbol{J}} = \begin{bmatrix} \boldsymbol{J}_{R-L} \\ \epsilon \begin{bmatrix} \boldsymbol{J}_R & 0 \end{bmatrix} \end{bmatrix}. \tag{20}$$

Figure 21 shows an example of NAO falling due to an impact push on its head and how the controller performs according to the proposed strategies with inertia shaping. The robot initially stands on both feet, and is subjected to a forward push which has an impulse of 2.5Ns. The four green columns are objects that the robot should avoid touching during the fall. Without a fall control, the robot collides with the front column as shown in Fig. 21(b). The latter figures show the effect of using different fall strategies. Simply lifting the left leg dramatically changes the fall direction as shown in Fig. 21(c-d). *Take a Step* strategy enables the robot to change the fall direction to the front right as shown in Fig. 21(e-f). The resolutions of the cell used in Eq. 7 are 10mm for (x, y) and 0.1 rad for β. The inertia shaping strategy can enhance the performance by creating angular momentum in order to indirectly modify the fall direction as shown in Fig. 21(g-h).

Fig. 22 contains two plots showing the convergence of two components of the overall angular velocity of the robot to their desired values, which is controlled according to Eq. 10. This control is achieved through the indirect control of the robot's locked inertia. The plots correspond to simulation shown in Fig. 21(g-h). The angular velocity components are shown to successfully converge to their desired values ω_d. The chattering like behavior of the angular velocity results from the back and forth motion of the CoP.

7 Experimental Results of Fall Direction Change

We have experimentally evaluated our fall controller in hardware using the Aldebaran NAO H25 robot. In this section we describe these experiments and also compare the experimental results to the simulation. Before we perform the experiments a few technical problems need to be solved.

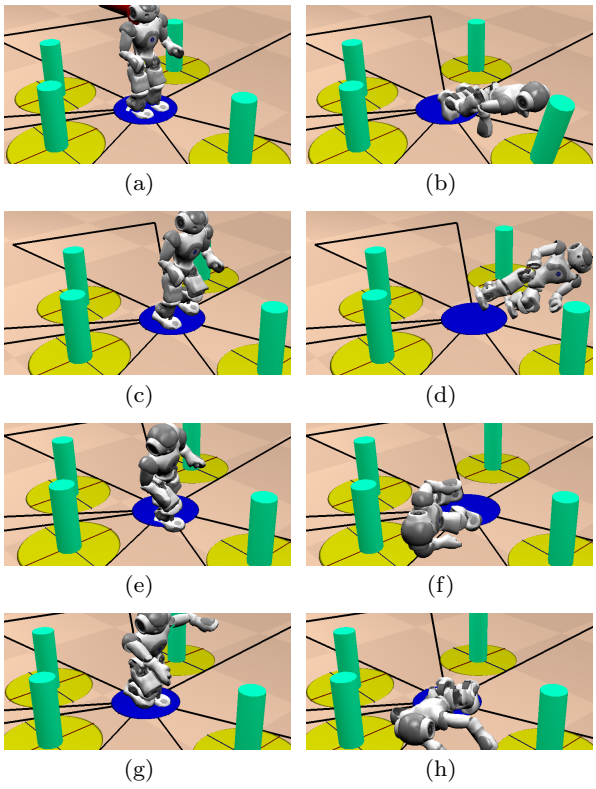


Fig. 21: Examples of direction-changing humanoid fall through the proposed control strategies. The NAO robot is surrounded by four green columns, which the robot should avoid hitting during a fall. The regions enclosed by the solid black lines and curves are safe cones. Planning of the controller takes place in the C-space. (a) The robot is pushed from behind by a 2.5Ns magnitude impulse. The thick red line depicts the push. (b) Without the fall controller, the robot ends up hitting the front object. (c) *Lift a Leg* strategy is used and the robot lifts the left leg so that the support area is modified. (d) The robot falls to the front left instead of the front. (e) The robot uses *Take a Step* strategy to change the fall direction. (f) The robot lands on the front right safe region. (g) Inertia shaping is used with the foot lifting strategy. (h) The robot falls to a safe region.

7.1 Estimation of global position and foot/ground contact point

The biggest challenge for the experiment is to estimate the global posture of the body frame while the robot is falling. In simulation, this information was readily available. However, during the experiment we have to estimate it using an IMU and force sensitive resistors (FSR) in the feet. The estimation is relatively easy when at least one foot has a firm contact with the ground. The relative 3D transformation between the foot and the body frame use its global position in a forward kinematics problem. However, when the robot is falling, its feet can lose the firm ground contact, and we need to

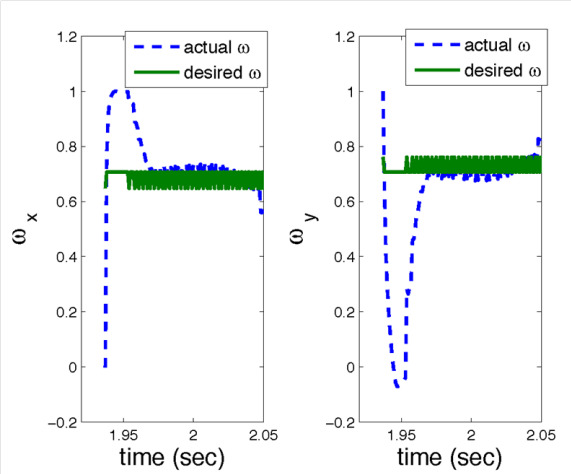


Fig. 22: The two plots show the effect of angular velocity control through momentum. The plots correspond to simulation run shown in Fig. 21(g-h). The two components of the normalized angular velocity are shown to successfully converge to their desired values ω_d. The chattering like behavior of the angular velocity is the result of back and forth motion of the CoP.

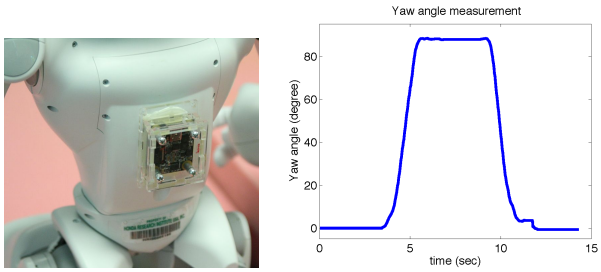


Fig. 23: (a) An additional IMU is attached on the back of the NAO robot. (b) Measured yaw angle when the robot is manually rotated by about 90° and returned to 0°.

depend on the IMU for the computation of the body coordinate frame.

Unfortunately, the built-in IMU in the NAO robot has only 2 gyroscopes and 3 accelerometers, which can compute only the roll and the pitch angles. The yaw angle of the robot would be missing. This may suffice when the robot has a firm contact on the ground, but not when foot toppling is involved. To estimate the missing yaw angle, we attached an additional IMU with 3 gyroscopes and 3 magnetometers externally on the robot's back, as shown in Fig. 23.

Note that even 3 accelerometers and 3 gyroscopes are not sufficient for a reliable estimation of the yaw angle; the 3 axis accelerometer gives only 2 reference angles for roll and pitch and the yaw angle estimation should solely rely on the gyro sensor information, which suffers from drift. From the additional IMU, the yaw angle is estimated using the planar heading of the sensor which is obtained from a magnetic compass output. Kalman filters [39] have been implemented for robust estimation of the roll and pitch angles.

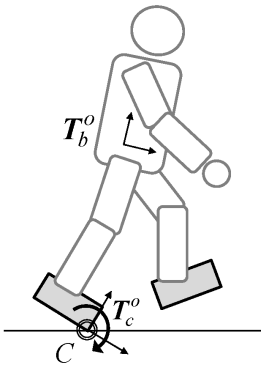


Fig. 24: Coordinates of the body and the anchor foot. C is the foot/ground contact point. $\boldsymbol{T}_c^0$ is a global frame of the contact point. and $\boldsymbol{T}_b^0$ is the global body frame. Note that the transformation from the foot to C with respect to the foot is constant.

Given these estimated orientations, the position of the body is still missing. In order to estimate it, we use the relative pose between the contact point and the body given the assumptions of no slip, no movement of contact point and a polygonal foot geometry. If the robot does not experience any slip and maintain the contact point during the fall, the contact point can be referenced for global position. Figure 24 shows an example of the falling robot. The point is that we can obtain the estimated orientation of the body frame (*i.e.*, body rotation matrix $\boldsymbol{R}_b^0$ in the global body transformation $\boldsymbol{T}_b^0$) and the estimated location of the foot contact point (*i.e.*, position column $\boldsymbol{P}_c^0$ in the global contact point transformation $\boldsymbol{T}_c^0$). Combining them results in the full posture of the body frame.

The following equation describes the relative posture between the contact point and the body frame:

$$\boldsymbol{T}_c^0 \boldsymbol{T}_b^c = \boldsymbol{T}_b^0, \tag{21}$$

which is equivalent to the following:

$$\begin{bmatrix} \boldsymbol{R}_c^0 & \boldsymbol{P}_c^0 \\ 0 & 1 \end{bmatrix} \boldsymbol{T}_b^c = \begin{bmatrix} \boldsymbol{R}_b^0 & \boldsymbol{P}_b^0 \\ 0 & 1 \end{bmatrix}, \tag{22}$$

where $\boldsymbol{T}$ is the transformation matrix and $\boldsymbol{R}$ and $\boldsymbol{P}$ are the rotation matrix and the position vector, respectively. The superscript and the subscripts 0, c and b refer to the global frame, the contact point frame and the body frame, respectively. In Eq. 22, the orientations of the contact point $\boldsymbol{R}_c^0$ and the position of the body frame $\boldsymbol{P}_b^0$ are unknown given the joint angles. Equa-

tion 22 can be rewritten as:

$$\boldsymbol{R}_c^0 = \boldsymbol{R}_b^0 \boldsymbol{R}_c^b \tag{23}$$

$$\boldsymbol{P}_b^0 = \boldsymbol{R}_c^0 \boldsymbol{P}_b^c + \boldsymbol{P}_c^0, \tag{24}$$

since

$$\boldsymbol{T}_b^c = \begin{bmatrix} \boldsymbol{R}_b^c & \boldsymbol{P}_b^c \\ 0 & 1 \end{bmatrix}. \tag{25}$$

In order to estimate the foot/ground contact point during fall, again we assume no slip and non-changing contact point during fall. Unlike during simulations where the contact point information could be directly obtained, we have to estimate it during the experiment. Since we also assume the foot area is a polygon, the contact point can be either an edge (the robot topples like a 2D inverted pendulum) or a vertex (the robot falls as a 3D inverted pendulum). We determine this from the 4 FSRs on each foot. At every control sampling time, the controller checks the values of the FSRs and sets on/off states of the sensors from a tuned threshold. From empirical data, the controller estimates that the contact is over an edge when two adjacent FSRs are *ON* and their values are equivalent while the other two are *OFF*. If one of them has a significantly higher value than the other, the controller interprets this a vertex contact.

A hardware implementation of the fall controller on the NAO robot must factor in its limited capabilities of sensing and control. The main differences between the simulation and the experiment are listed in Table 1. The strategies described in Section 5.4 are utilized.

7.2 Experimental results

A set of four experiments is designed to show the effect of the proposed strategies. The experiments start with *Lift a Leg* strategy, advance to *Inertia Shaping* with two different push directions, and show *Take a Step* strategy.

In the first experiment, the robot gets a steady push from behind until it switches to fall control mode and the proposed strategies are utilized. For repeatability, we use a linear actuator to give a push to the robot (the machine visible behind the NAO robot in Fig. 25). The controller runs on an external laptop connected to the robot via a wired network. The lean angle of the robot estimated from the IMU is used to trigger the direction-changing fall controller.

Without a fall controller, the robot falls forward as shown in Fig. 25. Figures 26(a-b) demonstrate that the *Lift a Leg* strategy can make a significant change under the same push. The robot lifts the right leg to change the fall direction and falls almost to the right.

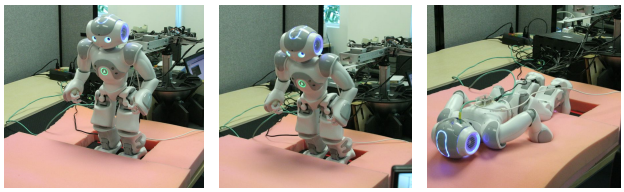
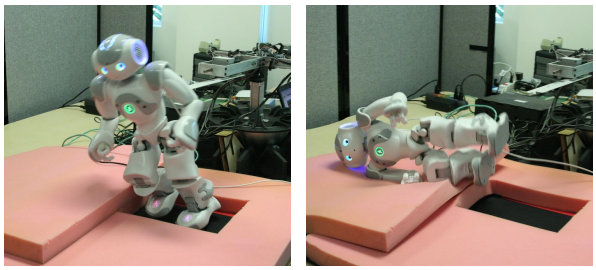


Fig. 25: Default case: without direction-changing fall controller, the NAO robot, when pushed from behind by a linear actuator, falls forward.



(a) (b)

Fig. 26: Snapshots of the fall experiment. Robot uses the *Lift a Leg* strategy. (a) The robot lifts up the right leg. (b) The robot falls almost completely to the right.

We tested two *Lift a Leg* strategies, one of which lifts the left leg and the other lifts the right leg, and the resultant CoM trajectories are compared in Fig. 27(a). According to Fig. 27(b), our fall controller seems to over-predict the resultant fall angles. We think that this difference is caused mainly due to the change of the foot/ground contact point during fall. The prediction comes from considering the robot as a 3D pendulum with a *fixed* contact point, which becomes invalid when the foot/ground contact point moves. For example, in Fig. 26(a), the foot/ground contact point is at the front-right corner of the left foot, which our controller correctly estimates and the predicted fall angle is 114°. However, somewhere between Fig. 26(a) and Fig. 26(b), the 3D fall motion of the robot causes the right edge of the left foot to become the foot/ground contact edge. This prevents the robot from rotating further backwards, and the robot ends up falling to the right (around 90°). In future work, this issue should be addressed in order to obtain better prediction accuracy.

In the next experiment, in order to test the effectiveness of inertia shaping, we performed an experiment to see if through inertia shaping we can cancel the effect of fall direction change, which was originally achieved through foot lifting. As seen in Fig. 26, lifting of the right foot causes the robot to fall toward its right. In the following example, after foot lifting, we execute inertia shaping using the forward direction fall as the objective. As seen in Fig. 28(a-b), inertia shaping

Simulation	Experiment
• Faster control sampling time (1 kHz)	• Slow control sampling time (≈30 Hz)
• Perfect knowledge of exact global position of the body frame	• Noisy estimation of global position of the body frame
• Perfect sensing of joint angle, velocity and acceleration	• Only joint angles sensed
• Perfect knowledge of foot/ground contact points	• Imperfect estimation of foot/ground contact point
• Polygonal feet	• Feet perimeter is curved
• Perfect knowledge of exact timing of push	• Timing of push is unknown

Table 1: Direction-changing Fall: Difference between simulation [26] and experiment

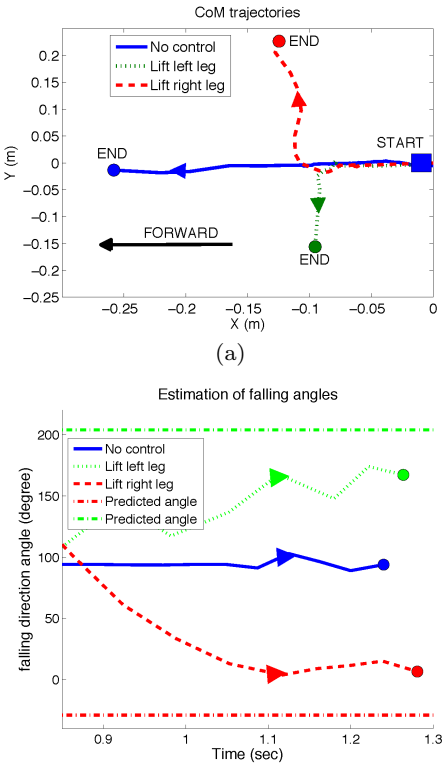


Fig. 27: (a) CoM trajectories of the robot during the *Lift a Leg* strategies. The circles denote the end of the trajectories. The solid blue curve is the CoM trajectory of the falling robot without a fall controller, and the dotted green and dashed red curves are trajectories from our fall controller by lifting the left and right leg, respectively. The forward direction is displayed by the black arrow. (b) Measured fall angles with respect to the *Lift a Leg* strategies and their estimations shown as horizontal lines at the top and bottom. The differences between estimations and experimental results mainly from the fact that the foot/ground contact point changes over time during fall.

is shown to have successfully canceled the effect of foot lifting and the robot falls forward. Note that the arms are stretched to maximize the effect of inertia shaping. In another inertia shaping experiment, we can make the robot fall diagonally, under the same forward push, as shown in Fig. 28(c-d). Figure 29 shows how inertia shaping changes the CoM trajectory.

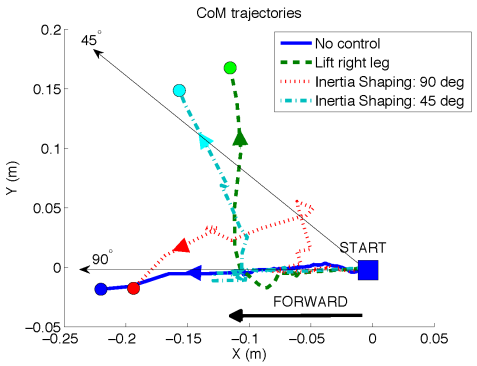


Fig. 29: CoM trajectories. The solid blue curve is the trajectory of the falling robot without any control. The dashed green curve corresponds to the *Lift a Leg* strategy. The dotted red curve is for the controller with inertia shaping with forward fall as the goal. The dot-dashed cyan curve is the result of inertia shaping with right forward fall as the goal. The circles denote the end of the trajectories.

The third experiment checks the effect of pure inertia shaping without involving any stepping. In this experiment, only inertia shaping is used to change the fall direction. In the experiment described in Fig. 28, the robot had very short time for inertia shaping because it spends a part of the fall time in lifting up a leg. In order to have more control time dedicated to inertia shaping, in this experiment we start from a single support pose of the robot as shown in Fig. 30(a). The robot is pushed from the left and falls to the right without inertia shaping. Two independent experiments of inertia shaping with 0° (forward) and 45° (forward right) desired fall angles are implemented. The success of this experiment is evident in the resultant CoM trajectories as shown in Fig. 30(b).

In the fourth experiment, Fig. 31 shows snapshots of the experiment for *Take a Step* strategy. A push comes from the left of the robot which is supported by the left foot only. The controller modifies the support area to change the fall direction to 45° (right forward). The support area changes from a rectangle to a line and then to a pentagon. The direction of fall changes, as expected, according to support area, and the resultant trajectory of the CoM is shown in Fig. 32 in which the

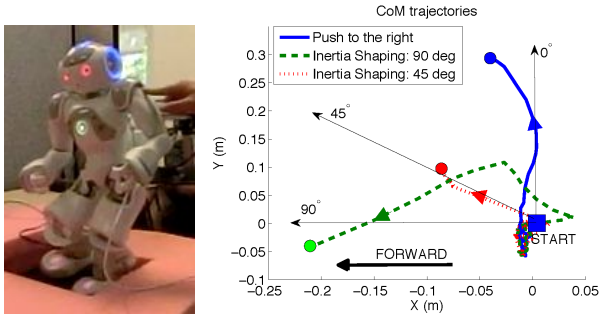


Fig. 30: Effect of inertia shaping during fall. (a) The NAO robot is in single support on the left leg when it is pushed from the right. (b) CoM trajectories with/without inertia shaping. Without inertia shaping, the robot falls to the right (solid blue curve). Two tests of inertia shaping with target angles of 45° (dotted red curve) and 90° (dashed green curve) are used to change the fall direction. The circles denote the end of the trajectories.

robot also takes a step to change the fall direction to −45° (right backward). When the humanoid is on single support in Fig. 31(a), it topples to the right and rotates about the right edge of the support foot as shown in Fig. 31(b). Once the robot takes a step with the right foot rotated by 45°, the support base extends to a pentagon as shown in Fig. 31(c). The direction of fall goes to the right forward since the reference point (capture point) is at the right forward of the support polygon.

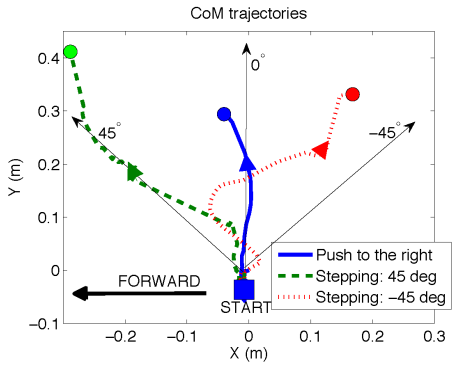


Fig. 32: CoM trajectories when the robot uses the *Take a Step* strategy. Two target falling angles (± 45°) are used. The solid blue line is the COM trajectory of the falling robot after a push from the left. The dashed green curve is from the *Take a Step* strategy with the 45° target angle, and the dotted red curve is for −45° target angle. The solid circles denote the end of the trajectories.

7.3 Comparison with simulation results

The bottom row figures of Fig. 28 show motions from simulations, which correspond to the experimental results with the same strategies and goals that are shown in the top row figures. In terms of the CoM trajectories, Fig. 33 shows the comparison between the trajectories. We see that the apparent motions in the experiments match well with those seen in the simulations though the specific states such as CoM trajectories are not identical.

In the experiment, we often encountered a problem due to lack of motor power in the experiments. Even though we use the same maximum joint speed and torque as in the simulations, motors subjected to high load often could not follow the desired trajectories and stalled as a consequence. During a fall, the robot is likely to be unbalanced and some joints would be under high gravitational load. Therefore, during fall control, the desired motion cannot be met since the joints cannot be actuated properly. In the experiment shown in Fig. 28(c-d), we found that the hip roll joint did not follow the desired trajectory, which caused a distorted motion and the resulting fall direction diverged from what was obtained in the simulation. This lack of power also makes it hard to achieve consistent results in the experiments. Given the same initial condition including a push, the robot may take the right action expected in simulation when every joint follows the desired trajectories but may not when any controlled joint is stalled. Thus, currently the capability of our fall controller is limited by the hardware specifications.

Also, the maximum rotational speed of the actuators did not match those that were obtained in the simulation. For example, the simple action of lifting-up a leg by the same height takes longer time in the experiment compared to that in the simulation. This means that we have smaller time to execute motions needed for the inertia shaping. The difference between the trajectories of Fig. 33 with 90° target angle comes from this speed limit. Since lifting-up takes more time in the experiment, inertia shaping starts working later in the experiment, and the moving-right CoM leads the lifted foot to touch the ground in the experiment before inertia shaping generates enough momentum to pull the CoM forward. This touch may leave a noisy trajectory due to the change of the contact point during fall. Note that we estimate the body posture based on the consistent contact point.

However, we can incorporate actual torque and velocity limits if we know them beforehand. Since the current NAO API does not support velocity control used in simulation, we had to modify the velocity controller

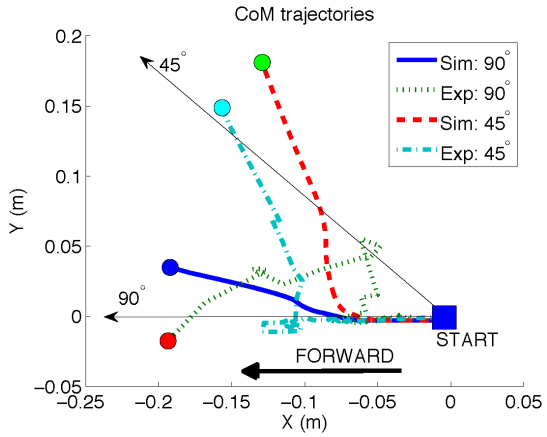


Fig. 33: CoM trajectories of the experiments and simulations in Fig 28. Lift a leg and Inertia shaping strategies are used. Two target falling angles (90° and 45°) are used. The solid blue curve is for the simulation with the 90° target angle, the dashed green curve is for the experiment with the 90° target angle, the dotted red curve is for the simulation with 45° target angle, and the dash-dotted cyan curve is for the experiment with the 45° target angle, The solid circles denote the end of the trajectories.

into a position controller. This controller modification and the slow control sampling time in the experiment (≈30 Hz) sometimes caused jerky motion.

8 Conclusion and Future Extensions

This paper reported the theory and hardware experiments of direction-changing fall control of humanoid robots among multiple objects. The fall controller contains a planner which, for a given set of surrounding objects, assigns merit scores to each direction around the robot. The scores depend on the position of the objects relative to the robot, and their sizes, which the robot is assumed to know all the time. The merit scores indicate the desired fall direction of the robot. The planner then logically evaluates the control strategies available to the robot and determines the best strategy or set of strategies to fall in the desired direction. The robot controller executes this strategy in an interactive manner such that real-time modifications can be made in case there is a risk of failure.

The robot employs two basic control strategies at its disposal, the foot placement control and the inertia shaping control. The foot placement control optimally changes the geometry of the foot support polygon of the robot in order to influence the evolution of the CoP on the polygon and indirectly control the fall direction. This strategy contains three components, No Action, Lift a Leg (left or right leg), and Take a Step (left or right step), all of which the planner evaluates separately. The inertia shaping control aims at appropriately modifying the global inertia of the robot, such that it possesses a desired angular momentum, thereby attaining the desired fall direction. The inertia shaping control can either recruit all the joints of the robot body, which is called Whole Body Inertia Shaping, or only those joints that are not used for foot placement, which is called Partial Inertia Shaping. The controller employed whole body inertia shaping when all other strategies were predicted to fail or when the selected strategy was sensed to leading to a failure. Several successful safe fall behaviors under a variety of external disturbances were demonstrated in simulation on a full-sized humanoid model and a smaller humanoid robot, the Aldebaran NAO H25. Hardware experiments on the NAO robot were also reported.

The theory and implementation of our direction-changing fall control have shown acceptable performance, however they also have revealed a few points for future discussion.

The planning procedure presented in this paper makes the following assumptions: (*i*) the fall direction estimated using an inverted pendulum approximation of the robot favorably compares with the actual fall direction (*ii*) all strategy executions are complete, *i.e.,* the robot is able to reach the desired configuration corresponding to the strategy before it touches the ground. Using more sophisticated models for better prediction of the terminal fall direction is one of the future avenues of this research. The falling motion of a robot is complex and it is hard to tightly control it because of underactuation. Estimation errors can accumulate very rapidly.

The second assumption is relevant for the performance of the fall controller. An example of an incomplete strategy is when the robot topples too far and hits the ground before the stepping controller gets enough time to extend the swing foot fully. This can happen at large inclination angle of the robot. In order to prevent this, the fall controller should select a plan that is based on a shorter execution time. However, we pay a price for this in terms of the performance of the fall controller. An incomplete optimal plan is not comparable to other plans, and can result in a complete failure.

A robot can be controlled to physically interact with objects in its surrounding during the fall and can advantageously modify the fall direction. This interaction can involve holding or pushing on a nearby wall or on a furniture. Also, the current work assumed that the objects surrounding the robot are stationary. This restriction can be relaxed and the algorithms updated to factor in moving objects such as humans.

Currently, the merit score based determination of the desired fall direction only considers the location and size of the objects with respect to the robot. There is no consideration either of the speed, the direction of movement of the robot or the direction of the disturbance force. Using these factors can refine the fall direction significantly. An important practical issue on the topic of fall control is the challenge of quantitative determination of the capabilities of the controller. This is necessary to provide a workable guideline for the robot user.

Most of the simulations and experiments are performed with the robot at the stationary upright condition. The main reason for selecting this posture is to introduce repeatability in the experiments and in the evaluation process. For example, we found it fair and repeatable to compare two fall control scenarios against a disturbance, both starting from the stationary upright initial pose. However, when the robot is walking, it is very difficult to compare the performances since it can vary greatly with very small differences in the initial robot states. Although we have tested a few isolated cases, our work on this important topic has not been exhaustive. The application of fall controllers on fully walking humanoids and their appropriate evaluation is one of the major topics for the future work. This study should also include the relationship between the walking speed and direction and the direction of fall.

An interesting avenue for future study is the effect of foot shape on the nature of fall. It is conjectured that small modifications in foot shapes can make it easier to change the robot's fall direction.

Since our experimental validation is limited to a small robot, implementing our fall controller on a full sized robot hardware will be an interesting future research topic. We actually expect better performance for a larger humanoid since the larger size allows a longer fall duration that gives more room for the fall controller such as a larger allowable stepping zone and longer execution of the inertia shaping control.

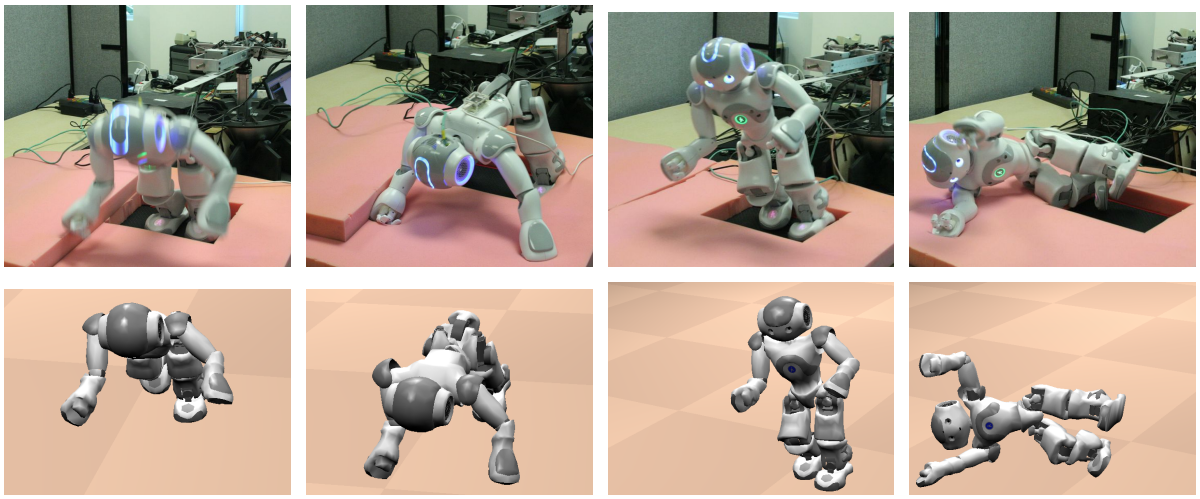
In this work we have introduced the concepts of fall trigger and control trigger. Developing robust fall trigger and control trigger would be essential, however it is extremely challenging since they are likely dependent on a number of variables such as a specific balance controller and the robot states.

Acknowledgements Seung-kook (2008), Umashankar (2009), Sung-Hee (2006), KangKang (2007), and Shivaram (2008) all contributed to this work during their internships at HRI at different times. Seung-kook (2010-2013) did major part of the subsequent work while working as a Senior Scientist at HRI.

References

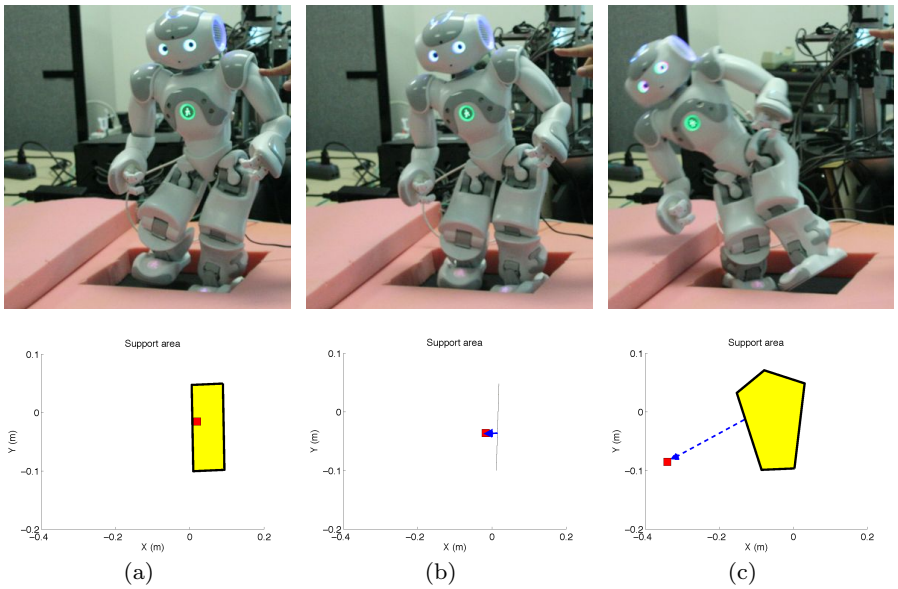
1. Ashton-Miller, J.A.: Biomechanics of mobility and fall-arrests in older adults. In: RESNA 2000: Technology for the new millenium, pp. 537–542. Orlando, Florida (2000)
2. Buss, S.R.: Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. Tech. rep., Department of Mathematics, University of California, San Diego (2004)
3. Chia, P.C., Lee, C.H., Chen, T.S., Kuo, C.H., Lee, M.Y., Chen, C.M.S.: Correlations of falling signals between biped robots and humans with 3-axis accelerometers. In: International Conference on System Science and Engineering, pp. 509–514. Macau, China (2011)
4. Cordero, A.F.: Human gait, stumble and ... fall? Ph.D. thesis, University of Twente, Enschede, The Netherlands (2003)
5. Cordero, A.F., Koopman, H., van der Helm, F.: Multiple-step strategies to recover from stumbling perturbations. Gait & Posture **18**(1), 47–59 (2003)
6. Cordero, A.F., Koopman, H., van der Helm, F.: Mechanical model of the recovery from stumbling. Biological Cybernetics **91**(4), 212–22 (2004)
7. DeGoede, K.M., Ashton-Miller, J.A.: Biomechanical simulations of forward fall arrests: effects of upper extremity arrest strategy, gender and aging-related declines in muscle strength. Journal of Biomechanics **36**, 413–420 (2003)
8. Fantoni, I., Lozano, R.: Non-linear Control for Underactuated Mechanical Systems (Communications and Control Engineering). Springer-Verlag London (2001)
9. Fujiwara, K., F., K., Saito, H., Kajita, S., Harada, K., Hirukawa, H.: Falling motion control of a humanoid robot trained by virtual supplementary tests. In: IEEE Intn'l Conf. on Robotics and Automation (ICRA), pp. 1077–1082. New Orleans, LA, USA (2004)
10. Fujiwara, K., Kajita, S., Harada, K., Kaneko, K., Morisawa, M., Kanehiro, F., Nakaoka, S., Harada, S., Hirukawa, H.: Towards an optimal falling motion for a humanoid robot. In: Humanoids06, pp. 524–529. Genova, Italy (2006)
11. Fujiwara, K., Kajita, S., Harada, K., Kaneko, K., Morisawa, M., Kanehiro, F., Nakaoka, S., Hirukawa, H.: An optimal planning of falling motions of a humanoid robot. In: IEEE/RSJ Intn'l Conf. on Intelligent Robots and Systems (IROS), pp. 456–462. San Diego, California, USA (2007)
12. Fujiwara, K., Kanehiro, F., Kajita, S., Hirukawa, H.: Safe knee landing of a human-size humanoid robot while falling forward. In: IEEE/RSJ Intn'l Conf. on Intelligent Robots and Systems (IROS), pp. 503–508. Sendai, Japan (2004)
13. Fujiwara, K., Kanehiro, F., Kajita, S., Kaneko, K., Yokoi, K., Hirukawa, H.: UKEMI: Falling motion control to minimize damage to biped humanoid robot. In: IEEE/RSJ Intn'l Conf. on Intelligent Robots and Systems (IROS), pp. 2521–2526. Lausanne, Switzerland (2002)
14. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: Mechatronic design of NAO humanoid. In: IEEE Intn'l Conf. on Robotics and Automation (ICRA), pp. 2124–2129. Kobe, Japan (2009)
15. Hobbelen, D.G.E., Wisse, M.: A disturbance rejection measure for limit cycle walkers: The gait sensitivity norm. IEEE Trans. on Robotics and Automation **23**(6), 1213–1224 (2007)
16. Höhn, O., Gačnik, J., Gerth, W.: Detection and Classification of Posture Instabilities of Bipedal Robots, pp.

409–416. Climbing and Walking Robots. Springer Berlin Heidelberg (2006)
17. Höhn, O., Gerth, W.: Probabilistic balance monitoring for bipedal robots. International Journal of Robotics Research **28**(2), 245–256 (2009)
18. Ishida, T., Kuroki, Y., Takahashi, T.: Analysis of motions of a small biped entertainment robot. In: IEEE/RSJ Intn'l Conf. on Intelligent Robots and Systems (IROS), pp. 142–147. Sendai, Japan (2004)
19. Kalyanakrishnan, S., Goswami, A.: Learning to predict humanoid fall. International Journal of Humanoid Robots **8**(2), 245–273 (2011)
20. Kanoi, R., Hartland, C.: Fall detections in humanoid walk patterns using reservoir computing based control architecture. In: 5th national conference on Control Architecture of Robots. Douai, France (2010)
21. Karssen, J.G.D., Wisse, M.: Fall detection in walking robots by multi-way principal component analysis. Robotica **27**(2), 249–257 (2009)
22. Kunihiro Ogata, Koji Terada, Yasuo Kuniyoshi: Falling motion control for humanoid robots while walking. In: Humanoids07, pp. 306–311. Pittsburgh (2007)
23. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge, UK (2006)
24. Lee, S.H., Goswami, A.: The reaction mass pendulum (RMP) model for humanoid robot gait and balance control. In: B. Choi (ed.) Humanoid Robots, pp. 167–186. In-Tech (www.intechweb.org), Austria (February, 2009). URL `http://intechweb.org/downloadpdf.php?id=6240`. (Earlier version appeared in ICRA 2007.)
25. Murray, R.M., Li, Z., Sastry, S.: A Mathematical Introduction to Robotic manipulation. CRC Press, Boca Raton (1994)
26. Nagarajan, U., Goswami, A.: Generalized direction changing fall control of humanoid robots among multiple objects. In: IEEE Intn'l Conf. on Robotics and Automation (ICRA), pp. 3316–3322. Anchorage, Alaska (2010)
27. Ogata, K., Terada, K., Kuniyoshi, Y.: Real-time selection and generation of fall damagae reduction actions for humanoid robots. In: Humanoids08, pp. 233–238. Daejeon, Korea (2008)
28. Pratt, J., Carff, J., Drakunov, S., Goswami, A.: Capture point: A step toward humanoid push recovery. In: Humanoids06. Genoa, Italy (2006)
29. Renner, R., Behnke, S.: Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes. In: IEEE/RSJ Intn'l Conf. on Intelligent Robots and Systems (IROS), pp. 2967–2973. Beijing (2006)
30. Robinovitch, S.N., Brumer, R., Maurer, J.: Effect of the "squat protective response" on impact velocity during backward falls. Journal of Biomechanics **37**(9), 1329–1337 (2004)
31. Robinovitch, S.R., Chiu, J., Sandler, R., Liu, Q.: Impact severity in self-initiated sits and falls associates with center-of-gravity excursion during descent. Journal of Biomechanics **33**, 863–870 (2000)
32. Robinovitch, S.R., Hsiao, E.T., Sandler, R., Cortez, J., Liu, Q., Paiment, G.D.: Prevention of falls and fall-related fractures through biomechanics. Exercise and Sports Sciences Review **28**(2), 74–79 (2000)
33. Ruiz-del Solar, J., Moya, J., Parra-Tsunekawa, I.: Fall detection and management in biped humanoid robots. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3323–3328 (2010)
34. del Solar, J.R., Palma-Amestoy, R., Marchant, R., Parra-Tsunekawa, I., Zegers, P.: Learning to fall: Designing low damage fall sequences for humanoid soccer robots. Robotics and Autonomous Systems **57**(8), 796–807 (2009)
35. Spong, M.W., Corke, P., Lozano, R.: Nonlinear control of inertia wheel pendulum. Automatica **37**, 1845–1851 (2001)
36. Stephens, B.: Humanoid push recovery. In: Humanoids07. Pittsburgh, PA, USA (2007)
37. Tan, J.S., Eng, J.J., Robinovitch, S.R., Warnick, B.: Wrist impact velocities are smaller in forward falls than backward falls from standing. Journal of Biomechanics **39**(10), 1804–1811 (2006)
38. Walker, M.W., Orin, D.: Efficient dynamic computer simulation of robotic mechanisms. ASME Journal of Dynamic Systems, Measurement, and Control **104**, 205–211 (Sept. 1982)
39. Welch, G., Bishop, G.: An introduction to the Kalman Filter. Tech. rep., Chapel Hill, NC, USA (1995)
40. Wilken, T., Missura, M., Behnke, S.: Designing falling motions for a humanoid soccer goalie. In: Proceedings of the 4th Workshop on Humanoid Soccer Robots (Humanoids 2009), pp. 79–84. Paris, France (2009)
41. Yin, K., Loken, K., van de Panne, M.: Simbicon: Simple biped locomotion control. ACM Transactions on Graphics **26**(3) (August 2007)
42. Yun, S.K., Goswami, A.: Humanoid robot safe fall using Aldebaran NAO. In: IEEE Intn'l Conf. on Robotics and Automation (ICRA), pp. 71–78. St. Paul, Minnesota, US (2012)
43. Yun, S.K., Goswami, A., Sakagami, Y.: Safe fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping. In: IEEE Intn'l Conf. on Robotics and Automation (ICRA), pp. 781–787. Kobe, Japan (2009)



(a) (b) (c) (d)

Fig. 28: (Top Row) Snapshots of the fall experiment with *Lift a Leg* strategy and inertia shaping. (Bottom Row) Snapshots of the fall simulations which have the same goal fall direction as the experiments. (a) After lifting up the right leg, the robot starts inertia shaping with the objective of canceling the effect of the *Lift a Leg* strategy. (b) Inertia shaping successfully makes the robot fall almost forward. (c-d) The robot uses inertia shaping to fall diagonally forward after lifting up the right foot, and inertia shaping is reasonably successful.



(a) (b) (c)

Fig. 31: The top row pictures are snapshots of falling humanoid with changing support polygon. The bottom row figures show the support polygon and Capture point. The small red square is Capture point. The dashed blue arrow is the estimated fall direction. (a) The support area is a rectangle formed by the left foot. Capture point resides inside the support area. (b) The robot is toppling after the push, and the support area in the inner edge of the left foot. Capture point is at the right, which implies the robot is falling to the right. (c) The robot has taken a step, and the support area is a pentagon formed by the contact points of the two feet. Capture point is out of the support area, and the robot falls diagonally as we intended. The target falling angle of the controller is 45° (right forward). The CoM trajectory of this experiment is shown in Fig. 32.