2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Safe Fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping

Seung-kook Yun
Massachusetts Institute of Technology
Cambridge, MA 02139
U.S.A.
yunsk@mit.edu

Ambarish Goswami
Honda Research Institute
Mountain View, CA 94041
U.S.A.
agoswami@honda-ri.com

Yoshiaki Sakagami
Honda Research Institute
Mountain View, CA 94041
U.S.A.
ysakagami@honda-ri.com

***Abstract*—Although fall is a rare event in the life of a humanoid robot, we must be prepared for it because its consequences are serious. In this paper we present a fall strategy which rapidly modifies the robot's fall direction in order to avoid hitting a person or an object in the vicinity. Our approach is based on the key observation that during "toppling" the rotational motion of a robot necessarily occurs at the leading edge or the leading corner of its support base polygon. To modify the fall direction the robot needs to change the position and orientation of this edge or corner vis-a-vis the prohibited direction. We achieve it through intelligent stepping as soon as a fall is detected. We compute the optimal stepping location which results in the safest fall. Additional improvement to the fall controller is achieved through *inertia shaping* techniques aimed at controlling the centroidal inertia of the robot.**

We demonstrate our results through the simulation of an Asimo-like humanoid robot. To our knowledge, this is the first implementation of a controller that attempts to change the fall direction of a humanoid robot.

***Index Terms*—humanoid robot fall, safe fall, fall direction change, support base geometry, inertia shaping**

I. INTRODUCTION

Safety is a primary concern that must be addressed before humanoid robots can freely exist in human surroundings. 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, to delicate and expensive objects in the surrounding or to a human being. Regardless of the substantial progress in humanoid robot balance control strategies, the possibility of a fall remains real, even unavoidable. Yet, a comprehensive study of humanoid fall and prescribed fall strategies are rare.

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.

A fall controller can target two major objectives independently or in combination: a) fall with a minimum damage and b) change fall direction such that the robot does not hit a certain object or person. The present paper introduces a strategy for fall direction change and describes a controller which can achieve both objectives.

Fig. 1 shows two cases of a fall caused by a frontward push on an upright standing humanoid robot (top figure). Without any fall controller, the robot falls forward and hits a block located in front of it (bottom, left). In the second case (bottom, right), the robot takes cognizance of the position of the block and the proposed controller successfully avoids hitting it.

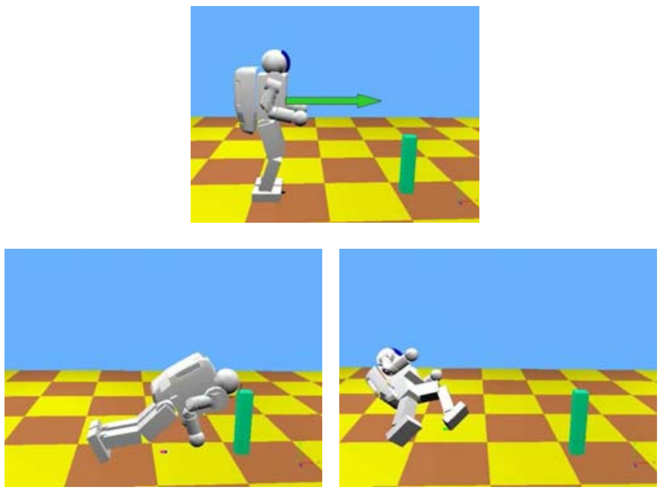
Fig. 1. Consequence of a humanoid fall without and with the proposed fall controller. The robot is initially in upright pose and is subjected to a forward push (top) shown by the green arrow. Without any fall controller the robot falls on the block object in front of it (bottom, left), potentially damaging or breaking it. For the same push, the fall controller successfully changes the fall direction and the robot is able to avoid hitting the object (bottom, right).

Let us note that *a fall controller is not a balance controller*. A fall controller complements, and does not replace, a balance controller. Only when the default balance controller has failed to stabilize the robot, the fall controller is activated. Further, *a fall controller is not a push-recovery controller*. A push-recovery controller is essentially a balance controller, which specifically deals with external disturbances of larger magnitude. A robot can recover from a push e.g., through an appropriate stepping strategy[12].

II. Background and Key Concepts

In this section we will review the existing literature and introduce some of the key concepts used throughout the paper.

A. Related work

Work on humanoid fall is rare. Fujiwara et al. at the Japan AIST Laboratory has done major work in the area of fall control of humanoid robots [7], [2], [3], [6], [4], [5]. Although these papers are concerned with the impact minimization problem, some important points of general applicability are established. [3] points out the advantages of a simulator for fall control research and [4] reports the design and building of a dedicated hardware (robot) for fall study.

In another work by Ogata et al., a fall detection condition based on the "degree of abnormality" to distinguish between fall and no-fall conditions was proposed [11], [10]. The robot improves fall detection through learning. We have also used a somewhat similar concept, which we call the *Fall Trigger Boundary* (FTB) and is described next.

B. Fall trigger boundary (FTB)

As shown in Fig. 2 the FTB encloses a region in the robot's feature space in which the balance controller is able to stabilize the robot. *An exit through the FTB is an indication of a certain fall* and this event is used to activate a switch from the robot's balance controller to a fall controller. The parameters that characterize the feature space can include both sensor data from and any number of computed variables such as center of mass (CoM) and center of pressure (CoP) positions, robot lean angle, angular momentum, etc. The shape and size of the FTB depends on the nature of the balance controller. Wieber [16] proposed a similar concept as viability kernel which tracks all the states as joint angles and velocities of a humanoid that adapts its motion according to the kernel. We focus not so much on the interior of the kernel but on the boundary between balanced and unbalanced regions.

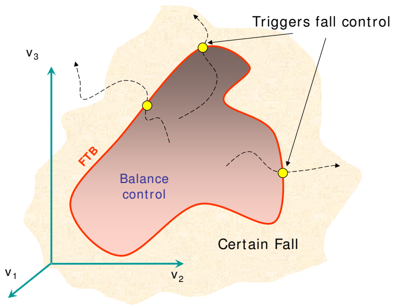

Fig. 2. Schematic of *Fall Trigger Boundary (FTB)*, a boundary in a humanoid feature space that surrounds the region where the humanoid is able to maintain balance. The axes in the figure represent different robot features such as CoM coordinates, angular momentum components, etc. The FTB represents the limit beyond which the robot controller must switch to a fall controller.

C. Support base geometry modification

The direction of fall of a humanoid robot is fully determined by the CoP location with respect to the support base. The support base 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 (or corner) of the support base. The robot rotates about this *leading* edge (corner). Therefore, *a change in the physical location of the leading edge (corner) of the support base with respect to the robot CoM exerts influence on the direction of robot rotation, i.e., the direction of fall.*

In Fig. 3 a humanoid robot is 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) of the support base and the robot will begin to rotate about this leading edge.

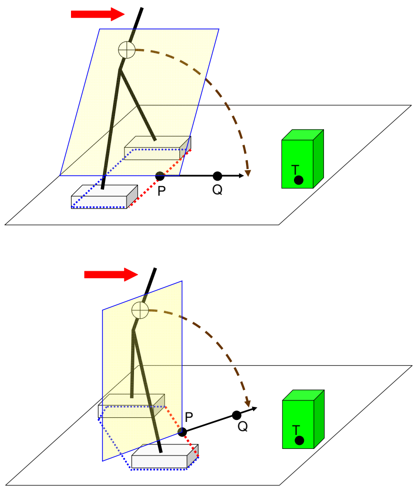

Fig. 3. A schematic diagram showing the basic idea behind fall direction change through support base geometry modification. A forward push is assumed. P denotes the CoP, and Q is the reference point (explained in text). The dotted lines show the support base (polygonal convex hull) of the robot while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

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 indicates the direction and magnitude of fall. In this paper we have used the *capture point*[12] as our reference point[1]. Although PQ may not be initially perpendicular to the leading edge of support base, 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 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).

There are two major challenges that we face. First, the robot becomes underactuated as soon as it starts toppling. This creates 1 or 3 additional dofs 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.

The proposed control strategy can be implemented according to the following steps:

[1]We will further describe capture point in Section III-A2.

1) Compute *control duration*, the length of time after the push force has disappeared, during which the controller is assumed to be active.
2) Estimate reference point location at the end of control duration, based on inverted pendulum model.
3) Compute optimal stepping location on the ground.
4) Control humanoid legs to step to optimal location.
5) Employ inertia shaping to generate angular momentum that further diverts the robot away from the obstacle.

III. Support Base Geometry Modification Controller

Once the humanoid state exits the fall trigger boundary, the fall controller estimates the direction and time-to-fall of the robot. Based on the estimation, it computes the best position to step and controls the leg accordingly.

The proposed approach sequentially employs two controllers, the support base geometry modification and inertia shaping, to change the direction of fall. This section describes the first controller.

A. Robot state estimation through simple humanoid models

To speed up calculations for predicting the robot states we approximate the robot with an equivalent inverted pendulum, see Fig. 4. 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 base, we model the robot with a 2D inverted pendulum. If instead, the CoP is located at a corner, the estimation uses a 3D spherical pendulum model. The 2D pendulum model has a closed-form solution. However, since the spherical 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.

1) Estimation of the control duration: Time-to-fall is a critical parameter for the evaluation and formulation of a fall response strategy. The biomechanics literature gives us a few data on the time-to-fall of human. A simple forward fall of an adult starting from a stationary 15° inclination takes about 0.98s, whereas that for a backward fall starting from stationary 5° inclination takes 0.749s (for a flexed knee fall) and 0.873s (for an extended knee fall)[14].

The fall controller remains active until the lean angle θ between the humanoid CoP-CoM line and the vertical axis crosses a certain threshold $\theta_{threshold}$. We assume that all external forces have disappeared when the robot starts to use the fall controller. The control duration is obtained through an incomplete elliptic integral of the first kind of the 2D pendulum model [13] when the lean angle goes over the threshold. For the spherical pendulum model, we simulate its dynamic equations.

2) Estimation of reference point: As mentioned before, we have used the capture point as the reference point in this work. Capture point is the point on the ground where a humanoid must step to in order to come to a complete stop after an external disturbance[12]. The location of the capture point is proportional to the linear velocity of the robot's CoM. Capture

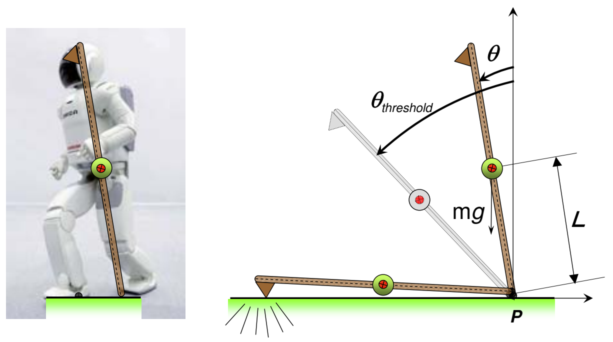

Fig. 4. Simple model of an inverted pendulum falling under gravity. P is CoP, m is the humanoid mass, and θ is the lean angle between the CoP-CoM line and the vertical. We use this model for the fast estimation of time duration and other parameters of the robot.

point (x_Q, y_Q) for an inverted pendulum approximation of the robot is computed as follows:

$$x_Q = x_G + \sqrt{\frac{z_G}{g}}\dot{x}_G \tag{1}$$

$$y_Q = y_G + \sqrt{\frac{z_G}{g}}\dot{y}_G \tag{2}$$

where (x_G, y_G, z_G) and $(\dot{x}_G,\, \dot{y}_G,\, \dot{z}_G = 0)$ are the robot CoM position and velocity, as estimated from the dynamic equations of the pendulum models.

Suppose the control duration is ΔT. In the 2D pendulum model, the velocity after ΔT is computed from the energy equation as follow:

$$\dot{\theta}(\Delta T) = \sqrt{\frac{2E}{I} - \frac{2mgL\cos(\theta(\Delta T))}{I}} \tag{3}$$

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

B. Definition of the optimal step location

Fig. 5 shows a robot that is about to topple. The old CoP P_1 has reached an edge of the support base, and the support base has shrunk to a line. Approximating the robot as a rigid body instantaneously, the trajectory of the CoM is parallel to P_1Q_1. T is the target object to avoid. Our goal is to find a point P_2 within the allowable stepping zone of the robot such that the robot is maximally diverted away from T, i.e., to maximize α_2.

Assuming that the humanoid is in double support phase, the optimal CoP is selected among the following 5 cases:

1) No change, i.e., robot does not react
2) (2 cases) Lifting (and not re-planting) left or right foot
3) (2 cases) Taking left or right step

We use a brute-force search for each case to find the optimal new CoP. The allowable stepping zone on the floor where the robot's swing foot can reach within the control duration is denoted by $\mathcal{D}$, shown as the green dotted polygon in Fig. 5. This area is divided into cells of x, y and β, the

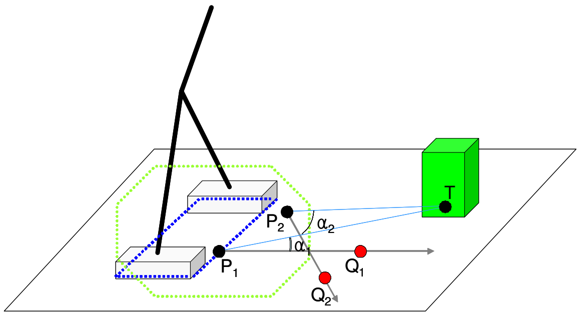

Fig. 5. Schematic of a biped robot subject to a fall. P_1 is the current CoP, and Q_1 is the reference point when the controller starts, and the robot will fall in the direction P_1Q_1. T is the target to avoid. The support base is shown in blue dotted lines. The green dotted lines enclose the allowable stepping zone, the region where a new CoP can be placed through proper foot placement. P_2 is a candidate for the new CoP by stepping. Q_2 is the estimated new reference point when the step is taken. α_i is the avoidance angle between P_iQ_i and P_iT. The fall controller will try to maximize α_2.

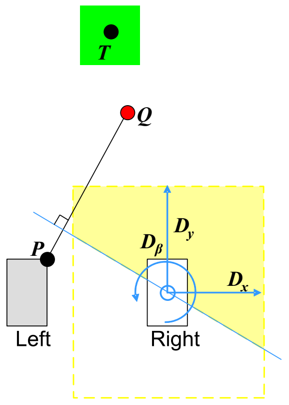

Fig. 6. In this figure the allowable stepping zone is shown in yellow. The left foot is the support foot. P is the CoP with the single support and Q is the reference point. The allowable stepping zone is the upper part of the rectangle (above the blue separatrix) with edges D_x and D_y. D_β denotes amount of rotation of the swing foot.

angular displacement (of foot). We re-locate the non-support foot according to (x, y, β) of each cell, and estimate a new reference point and CoP.

The avoidance angle α is computed for each case and the optimal CoP is selected as follows:

$$P_2 = \underset{P_2 \in CoP(\mathcal{D})}{\operatorname{argmax}} \ angle(Q_2P_2T) \quad (4)$$

We assume a rectangular 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.

C. Estimation of the allowable stepping zone

Given the control duration, the allowable stepping zone is estimated using leg Jacobians. Suppose the robot has a firm support on the ground. With two legs, we have the following equations:

$$\dot{\boldsymbol{P}}_L - \dot{\boldsymbol{P}}_{body} = \boldsymbol{J}_L\dot{\boldsymbol{\theta}}_L \quad (5)$$

$$\dot{\boldsymbol{P}}_R - \dot{\boldsymbol{P}}_{body} = \boldsymbol{J}_R\dot{\boldsymbol{\theta}}_R, \quad (6)$$

where $\boldsymbol{P}_L$ and $\boldsymbol{P}_R$ are the positions of the left and right feet, respectively, with respect to the support foot $\boldsymbol{P}_L$, and $\boldsymbol{P}_{body}$ is the location of the body frame, $\boldsymbol{\theta}_L$ and $\boldsymbol{\theta}_R$ are 6×1 joint angle vectors of the robot legs, and $\boldsymbol{J}_L$ and $\boldsymbol{J}_R$ are the leg Jacobian matrices.

Performing (Eq. 6)-(Eq. 5):

$$\dot{\boldsymbol{P}}_R - \dot{\boldsymbol{P}}_L = [\boldsymbol{J}_R \ \ -\boldsymbol{J}_L] \left[\dot{\boldsymbol{\theta}}_R \ \dot{\boldsymbol{\theta}}_L\right]^T \quad (7)$$

where we have used the (6×12) foot-to-foot Jacobian matrix as $\hat{\boldsymbol{J}} = [\boldsymbol{J}_R \quad -\boldsymbol{J}_L]$.

The size of the allowable stepping zone is estimated by the Jacobian as shown in Fig. 6.

$$D_{k(x,y,\beta)} = \Delta T \sum_{i=1}^{12} \left| \hat{\boldsymbol{J}}_{ki}\dot{\theta}_i^{MAX} \right| \approx \gamma \Delta T \sum_{i=1}^{12} \left| \hat{\boldsymbol{J}}_{ki} \right| \quad (8)$$

where $\dot{\theta}_i^{MAX}$ is the maximum velocity of a leg joint. γ is a constant included to approximate $\dot{\theta}_i^{MAX}$, which is assumed same for all joints.

We use only the upper half of the region cut by the separatrix line which is perpendicular to PQ and goes through the center of the moving foot. This is because a robot that is falling along PQ can hardly place its foot on the other side.

D. Step controller for a toppling humanoid

Once the optimal step location is computed one can expect to simply control the joint angles through an invers kinematics. However, taking a successful step to the optimal step location is not trivial because inverse kinematics solution will not be precise for a toppling robot. The main problem is that the support foot of the robot is not flat with the ground, i.e., it is underactuated, and robot is not likely to step as expected.

To compensate for this we need to know the foot rotation angle of the robot. Assuming that the robot possesses a gyro sensor in the trunk, the foot rotation angle can be estimated by noting the mismatch between the trunk orientation angle as computed by the gyro 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 fall, the CoP is modeled as passive rotational joint at which the support foot rotates, as shown in Fig. 7.

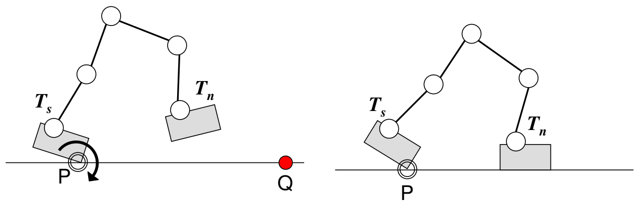

Fig. 7. (Left) For a robot that is falling, the support foot rotates about the CoP, P, towards the reference point Q. We model a free joint at P. Without external forces, the joint angle should increase. (right) desired landing posture

Although we cannot actuate the passive joint, the support foot is expected to rotate as estimated in the pendulum models *without* control. A transformation matrix from the support foot to the non-support foot $\boldsymbol{T}_n^s$ is:

$$\boldsymbol{T}_n^s = \boldsymbol{T}_s^{0^{-1}} \boldsymbol{T}_n^0 \tag{9}$$

where $\boldsymbol{T}_n^0$ is the transformation of the non-support foot and already fixed in optimizing the CoP.

If the joints are controlled according to $\boldsymbol{T}_n^s$ before the non-support foot hits the ground as in Fig. 7 (Left), the robot is expected to step on the desired location by gravity, Fig. 7 (Right). Note that the pre-computed $\boldsymbol{T}_s^0$ is for the landed posture, Fig. 7(b).

We can compute joint velocities to move the swing leg as.

$$\dot{\theta} = \hat{\boldsymbol{J}}^{\#}(\dot{\boldsymbol{P}}_R - \dot{\boldsymbol{P}}_L) \tag{10}$$

where $\hat{\boldsymbol{J}}^{\#}$ is the least damped-square pseudo-inverse of $\hat{\boldsymbol{J}}$.

IV. Whole-body falling motion control through inertia shaping

The humanoid can attempt to change the fall direction further, after the step is taken. Since a falling robot is normally underactuated, direct control of the CoM would not be effective. However, we can indirectly change the fall direction by generating angular momentum against the direction to the target. For this we can employ the inertia shaping technique [8]

In inertia shaping we control the centroidal composite rigid body (CRB) inertia [15] or the locked-inertia of the robot. Cenroidal CRB inertia is the instantaneous rotational inertia of the robot if all its joints are locked. Unlike linear inertia the CRB inertia is a function of the robot configuration and varies continuously.

Approximating the robot as a reaction mass pendulum, RMP[8], or an inverted pendulum with inertial mass, and assuming no slip at the ground, its CoM velocity can be computed as (see Fig. 8):

$$\boldsymbol{V}_G = \omega_G^P \times \boldsymbol{PG} \tag{11}$$

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

$$\omega_d = -\boldsymbol{e}_{\boldsymbol{z}\times\boldsymbol{PT}}, \tag{12}$$

where $\mathbf{e}_{\boldsymbol{z}\times PT}$ is a unit vector along the cross product between by $\boldsymbol{z}$ and $\boldsymbol{PT}$. The desired locked inertia is obtained as $\boldsymbol{I}_d = \boldsymbol{RIR}^{-1}$, where $\boldsymbol{I}$ is the current locked inertia and $\boldsymbol{R}$ is the rotation matrix obtained with an exponential map[9] from 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.,

$$\delta^s\widehat{\boldsymbol{I}} = \boldsymbol{J}_{\boldsymbol{I}}\delta\boldsymbol{\theta}. \tag{13}$$

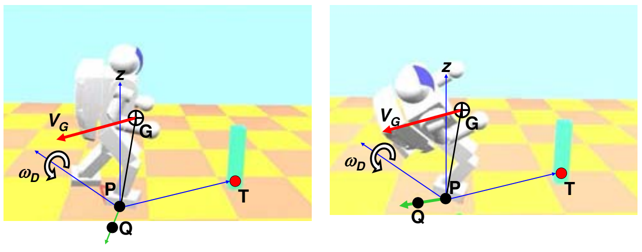

Fig. 8. To avoid falling on the block, V_G, the linear velocity of the robot CoM should be away from T. For this the robot should overall rotate around an axis obtained by the cross product of $\boldsymbol{PT}$ and the vertical, where P is the CoP.

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

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}_I^{\#}(\boldsymbol{I}_d - \boldsymbol{I}) \tag{14}$$

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 enough to obtain the desired V_G, however, even a modest change is very useful.

V. Simulation Results

We have performed the simulations using Webots [1], a commercial mobile robot simulation software developed by Cyberbotics Ltd. The humanoid fall is simulated with an sharp push of small duration. We have tested two initial conditions of the humanoid: standing and walking.

A. Standing humanoid

The humanoid stands on both feet, and is subjected to a push on its trunk for 0.1s. The push has a magnitude of 200N forward and 50N to the right. The target for the humanoid to avoid is located 1.2m in front of it, and the head of the humanoid would hit it without control. The fall controller starts 0.05s after the push has ended. Inertia shaping, if used, begins to work as soon as the swinging foot contacts the ground. We present results with support base change only and with both of inertia shaping and support base change.

Fig. 9 shows snapshots of the simulation. The support base changes from a rectangle to a point, then to a quadrilateral and back to a rectangle. 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 (9(b)). Taking a step makes the support base polygon a quadrilateral (9(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.

Fig. 10 shows the motion of a falling humanoid with both the support base geometry controller and inertia shaping controller. After taking a step (10(b)), the humanoid appears to change the falling direction by rolling the upper body backward (10(c)).

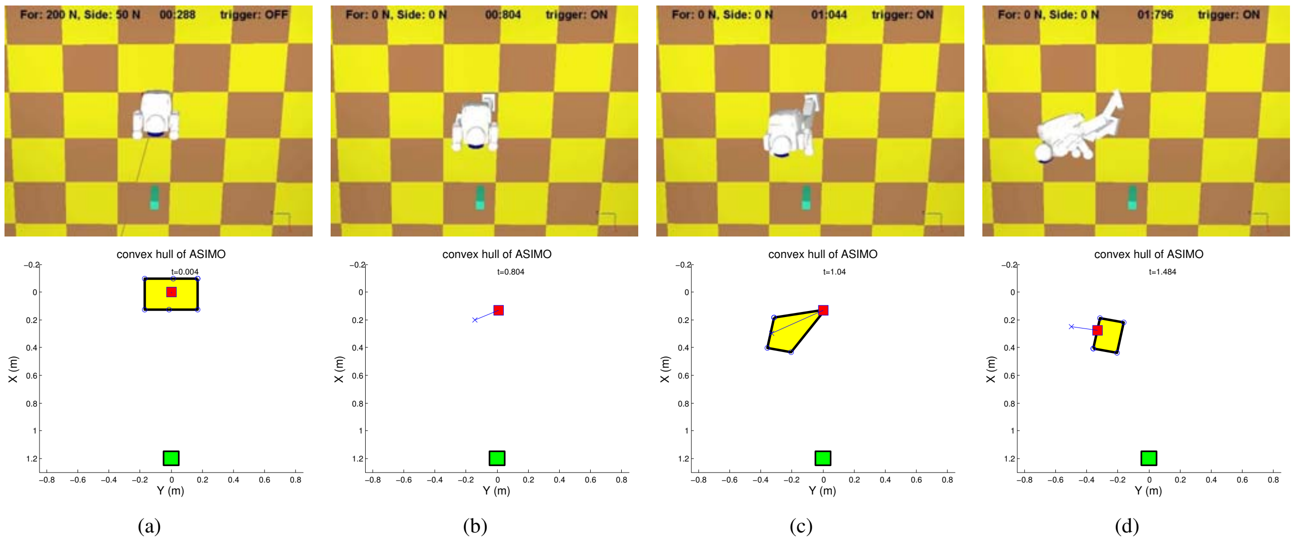

Fig. 9. Upper pictures are the top views of falling humanoid with changing support polygon. The lower figures show the support polygon and significant points. The yellow region is the support polygon. The green square is the target object (to avoid). The small red square is CoP, and the cross mark is the reference point. These two points are connected by the blue line to show the estimated direction of fall. (a) The humanoid gets a forward push. Direction of the push is shown as the red line. The support polygon is a rectangle formed by the two feet. (b) The humanoid has lifted the right foot to take a step. The 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 three points of the right foot and one point of the left foot. (d) The humanoid is falling leftwards, rotating about the rightmost edge of the right foot.

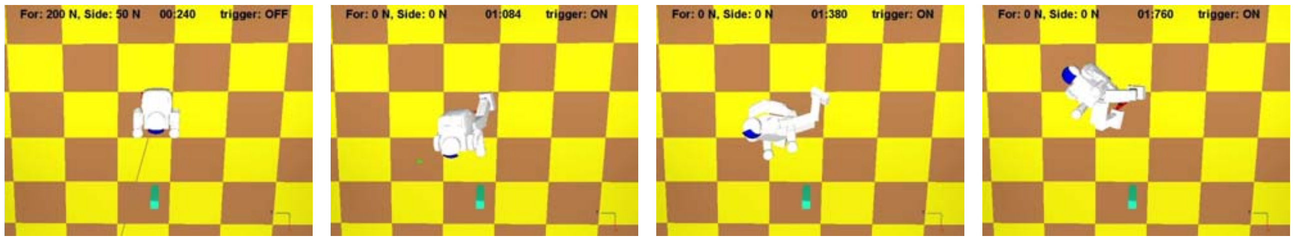
For: 200 N, Side: 50 N 00:240 trigger: OFF
For: 0 N, Side: 0 N 01:084 trigger: ON
For: 0 N, Side: 0 N 01:380 trigger: ON
For: 0 N, Side: 0 N 01:760 trigger: ON

Fig. 10. Snapshots of falling humanoid which uses both the support base geometry controller and inertia shaping controller. The push is the same as in Fig. 9. Inertia shaping starts after taking a step in the third picture. The humanoid appears to lean its body backwards as if it does limbo. After inertia shaping, the humanoid has fallen almost backwards.

Comparison of the CoM trajectories for the cases - no control, support polygon change, and support polygon change plus inertia shaping - are shown in Fig. 11. The figure clearly shows that the trajectory of the full controller diverges from the trajectory of the support polygon change and goes backwards.

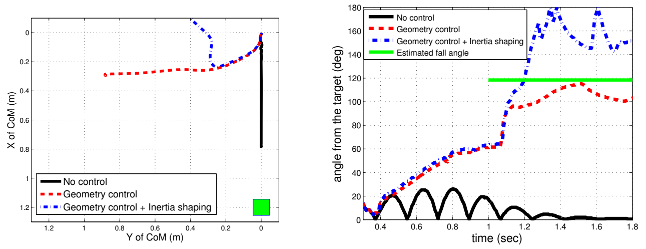

Fig. 11. Simulation plots of CoM trajectories (Left) and avoidance angles (Right) of a falling humanoid which was pushes during upright standing. The humanoid falls on the target without any control, which corresponds to a 0° avoidance angle. Intelligence footstep control improves this to 100° and inertia shaping further improves to 180°.

B. Walking Humanoid

The humanoid is subjected to a forward push during forward walking. The push is 150N for 0.1 second. Snapshots of the simulation are shown in Fig. 12. After the push, the humanoid takes a step on the front right of the ground by the swinging right leg, and the direction of fall changes to left forward. Inertia shaping yields a larger angle of falling direction from the target as shown in Fig. 13.

VI. Discussion and Conclusions

We have presented a humanoid robot fall controller, the objective of which is to rapidly modify the fall direction in order to avoid hitting an object or a person in the vicinity. The approach taken by us is to modify the support base geometry of the robot which indirectly, but strongly, modifies the fall direction of the robot. The shape of the support base should be carefully adjusted such that it does not possess any leading edge facing the object to avoid.

We have implemented the controller through an intelligent foot placement strategy of the robot, which triggers as soon as a fall is detected. We have shown how an optimal stepping

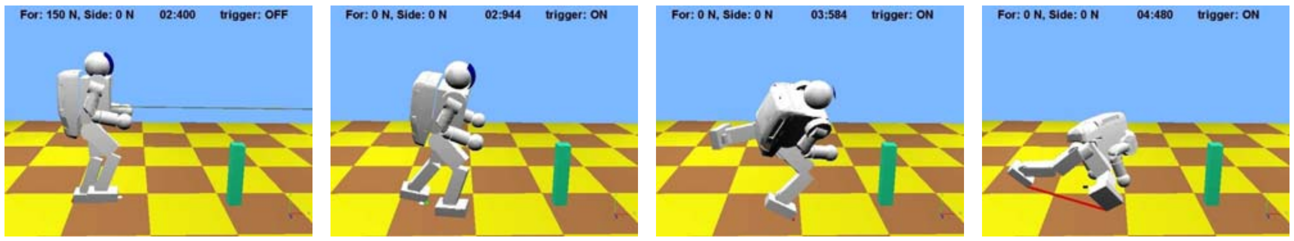
For: 150 N, Side: 0 N 02:400 trigger: OFF
For: 0 N, Side: 0 N 02:944 trigger: ON
For: 0 N, Side: 0 N 03:584 trigger: ON
For: 0 N, Side: 0 N 04:480 trigger: ON

Fig. 12. Humanoid gets a forward push while walking. It was supposed to take a step forward, however it has changed the plan and steps right-forward to change the direction of fall. After the step, it does inertia shaping and tries to move away from the target. Inertia shaping is not efficient enough to change the direction completely backwards, however the direction deviates a little more compared to a case of changing the support polygon only.

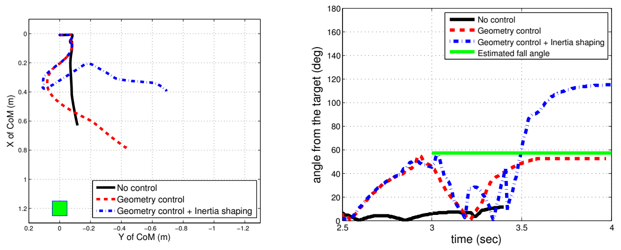

Fig. 13. Simulation plots of CoM trajectories (Left) and avoidance angles (Right) of a falling humanoid which was pushed while walking. Intelligence footstep control results in a 50° avoidance angle and inertia shaping further improves to 120°.

location can be computed, one that maximally diverts the avoidance angle, the angle between the fall direction and the direction to avoid. Additionally, we have applied inertia shaping controller to further divert the robot. We have demonstrated our results through the simulation of an Asimo-like humanoid robot.

To our knowledge, this is the first implementation of a controller that attempts to change the fall direction of a humanoid robot.

Falling is an unstable motion in nature, and it is hard to tightly control it. Estimation errors can accumulate in our method especially because we have used approximate inverted pendulum models for predicting the robot states at a future time. Currently, our controller has two distinct phases including the modification of the support base polygon and inertia shaping. However, they can be blended together in a control scheme.

ACKNOWLEDGMENTS

Seung-kook was supported by a Honda Research Institute USA summer internship. Important help from Dr. Kankang Yin as a summer intern is also gratefully acknowledged.

REFERENCES

[1] Cyberbotics. Webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.

[2] K. Fujiwara, Kanehiro F., S. Kajita, K. Yokoi, H. Saito, K. Harada, K. Kaneko, and H. Hirukawa. The first human-size humanoid that can fall over safely and stand-up again. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1920–1916, 2003, Las Vegas, NV, USA.

[3] K. Fujiwara, Kanehiro F., H. Saito, S. Kajita, K. Harada, and H. Hirukawa. Falling motion control of a humanoid robot trained by virtual supplementary tests. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1077–1082, 2004, New Orleans, LA, USA.

[4] K. Fujiwara, S. Kajita, K. Harada, K. Kaneko, M. Morisawa, F. Kanehiro, S. Nakaoka, S. Harada, and H. Hirukawa. Towards an optimal falling motion for a humanoid robot. In *Proceedings of the International Conference on Humanoid Robots*, pages 524–529, 2006.

[5] K. Fujiwara, S. Kajita, K. Harada, K. Kaneko, M. Morisawa, F. Kanehiro, S. Nakaoka, S. Harada, and H. Hirukawa. An optimal planning of falling motions of a humanoid robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 524–529, 2007.

[6] K. Fujiwara, F. Kanehiro, S. Kajita, and H. Hirukawa. Safe knee landing of a human-size humanoid robot while falling forward. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 503–508, September 28– October 2 2004, Sendai, Japan.

[7] K. Fujiwara, F. Kanehiro, S. Kajita, K. Kaneko, K. Yokoi, and H. Hirukawa. UKEMI: Falling motion control to minimize damage to biped humanoid robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2521–2526, September 30 – October 4, 2002 Lausanne, Switzerland.

[8] S-H. Lee and A. Goswami. Reaction mass pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4667–4672, April 2007.

[9] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic manipulation*. CRC Press, Boca Raton, 1994.

[10] K. Ogata, K. Terada, and Y. Kuniyoshi. Real-time selection and generation of fall damagae reduction actions for humanoid robots. In *Humanoids08*, pages 233–238, Dec. -3 2008, Daejeon, Korea.

[11] K. Ogata, K. Terada, and Y. Kuniyoshi. Falling motion control for humanoid robots while walking. In *IEEE-RAS 7th International Conference on Humanoid Robots*, Pittsburgh, 2007.

[12] J. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture point: A step toward humanoid push recovery. In *Humanoids06*, December, Genoa, Italy 2006.

[13] M. W. Spong, P. Corke, and R. Lozano. Nonlinear control of inertia wheel pendulum. *Automatica*, 37:1845–1851, February 2001.

[14] J-S. Tan, J. J. Eng, S. R. Robinovitch, and B. Warnick. Wrist impact velocities are smaller in forward falls than backward falls from standing. 39(10):1804–1811, 2006.

[15] M. W. Walker and D. Orin. Efficient dynamic computer simulation of robotic mechanisms. *ASME Journal of Dynamic Systems, Measurement, and Control*, 104:205–211, Sept. 1982.

[16] P.-B. Wieber. On the stability of walking systems. In *Proceedings of the International Humanoid and Human Friendly Robots*, 2002.