IEEE ICRA 2014, Hongkong

Tripod Fall: Concept and Experiments of a Novel Approach to Humanoid Robot Fall Damage Reduction

Seung-kook Yun and Ambarish Goswami

***Abstract*— This paper addresses a new control strategy to reduce the damage to a humanoid robot during a fall. Instead of following the traditional approach of finding a favorable configuration with which to fall to the ground, this method attempts to stop the robot from falling all the way to the ground. This prevents the full transfer of the robot's potential energy to kinetic energy, and consequently results in a milder impact. The controlled motion of the falling robot involves a sequence of three deliberate contacts to the ground with the swing foot and two hands, in that order. In the final configuration the robot's center of mass (CoM) remains relatively high from the floor and the robot has a relatively stable three-point contact with the ground; hence the name *tripod fall*. The optimal location of the three contacts are learned through reinforcement learning algorithm. The controller is simulated on a full size humanoid, and experimentally tested on the NAO humanoid robot. In this work we apply our fall controller only to a forward fall.**

I. INTRODUCTION

Safety is one of the main concerns behind the independent autonomous existence of humanoid robots in physically interactive human environments. Although the loss of balance and fall are rare in isolated or controlled environments, the situation would be different when the physical separation between robots and humans gradually disappears. A fall of a humanoid robot is a particularly worrisome event; a fall from an upright posture can cause damage to the robot, to delicate and expensive objects in the surrounding or can inflict injury to a nearby human being. Regardless of the substantial progress in humanoid robot balance control strategies, the possibility of fall remains real, even unavoidable.

A humanoid fall may be caused by external factors such as unexpected or excessive forces, unusual or unknown floor slipperiness, slope or profile of the ground, causing the robot to slip, trip or topple. Falls can also result from internal factors such as a malfunction or a failure of actuator, power or communication system where the balance controller is partially or fully incapacitated. In this paper we consider only those situations in which a fall is caused by an external factor; in particular the sensors and the motor power must remain intact such that the robot can execute a prescribed control strategy.

A falling humanoid can be easily appreciated as a formidable system to study and control. During a fall, a humanoid behaves as a nonlinear underactuated multi-body system with high-DOF possessing a continuously changing morphology and which must interact with unpredictable contacts. Additionally, the robot is subjected to the relentless gravitational pull, which it cannot fully resist, and which tries to bring it down to a crash. Consequently, the time interval between the detection of a fall and the actual event of ground impact is very short. Yet, through simulation and experiments we are able to demonstrate that meaningful modification to the default fall behavior can be achieved and damage can be reduced.

Seung-kook Yun (seungkook.yun@sri.com) is with SRI International, 333 Ravenswood Ave, Menlo Park, CA, USA. Ambarish Goswami (agoswami@honda-ri.com) is with Honda Research Institute US., 425 National Ave, Suite 100, Mountain View, CA, USA. This work was done at Honda Research Institute.

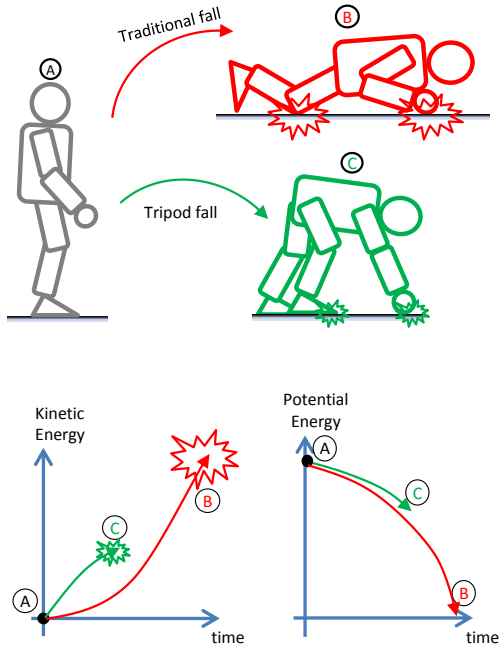


Fig. 1. The main idea of the tripod fall controller: (A) Imagine an upright standing humanoid robot on flat ground starting to toppling forward. (B) In traditional fall the humanoid CoM comes close to the ground resulting in high kinetic energy which gets converted from the high potential energy of (A). This causes high impact velocity. (C) The tripod fall controller stops the CoM at a height from the floor so that a smaller amount of kinetic energy gets accumulated. Thus, impact velocity is reduced. In addition, stepping absorbs part of the kinetic energy.

The tripod fall strategy, which is presented in this paper, aims at reducing the damage to the humanoid by preventing it from falling all the way to the ground. This is inspired by acknowledging that the high speed of impact at the end of a fall results from the large accumulation of kinetic energy during the fall. Therefore, the earlier we can arrest the humanoid motion, the more we may reduce the impact velocity and consequently the impact force, as schematically shown in Fig. 1.

Fig. 1 schematically describes the tripod fall strategy in terms of potential energy (PE) and kinetic energy (KE) of the

falling humanoid. Starting from the initial upright standing posture denoted by A in Fig 1 virtually all the previous research on damage reducing fall accepted the inevitability of posture B of Fig. 1, which corresponds to a *complete fall* to the ground. These strategies tried to minimize the impact velocity either by motion optimization [1], [2] or through heuristics [3], [4], [5]. Also in contrast to earlier works we use a stepping motion to break the fall early. In other words, the robot's first contact with the ground after the start of the fall is through the stepping foot rather than through knee or hand.

The tripod fall controller is designed to prevent the fall as early as possiblewhen the robot CoM is high from ground as shown in C of Fig 1. In this posture the humanoid is supported by three contact points, one foot and two hands, mimicking a tripod.

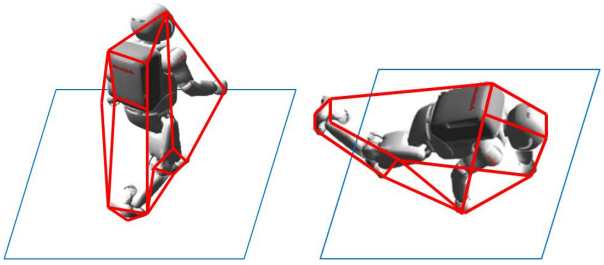
Compared to the previous works which used the knees of the *combined* legs as the first contact point (assuming the robot is falling forward) [1], [2], our controller uses a *stepping location* of the swing foot as the first contact. Therefore the stepping location can be anywhere on the ground that the swing foot can reach during fall, while the previous approaches had fixed first contact since the length from the foot to the knee is constant. In other words, our controller can actively choose the first contact point, not passively accepting the fixed first contact. In addition, using the swing foot as the first contact point keeps the CoM higher than using the knee for the first contact.

In the current implementation of tripod fall, the arms are fully stretched out for simplicity such that the controller needs to tune only the shoulder angles while the elbow and the wrist are locked. In future, impedance control will be used for the arm contact in order to further reduce the damage to the hand and the arm.

II. TRIPOD FALL CONTROLLER

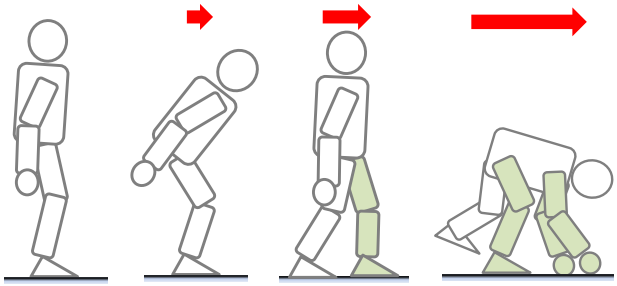
The tripod fall control can be generalized as the morphing of a rolling polyhedra as shown in Fig. 2. Viewing a humanoid as a 3D convex hull composed of all the outer vertices, edges and faces (ignoring curved surfaces, if any), the tripod fall control can be thought of as changing the polyhedra so that the robot performs a planned roll with the three contacts by changing its joint angles. In this paper, we use only one foot and two hands, however we can generalize this concept of rolling polyhedra so that the controller morphs the 3D convex hull consecutively so that the polyhedra rolls as the controller plans. For example, a complete 360-degree forward rolling motion such as in Japanese martial art technique called Ukemi [5] may be achieved by modeling the humanoid as the corresponding polyhedra and controlling it to roll over the ground until the two feet lands on the ground again after rolling. Implementation of such concepts as rolling over, falling to the shoulder side and falling backward remains future work.

The important parameters for the tripod fall are the stepping location and the shoulder angles. Due to the highly nonlinear dynamics of a falling humanoid, a learning algorithm is used to find the best parameters using multiple trials.



(a) (b)

Fig. 2. The generalized concept of humanoid fall as a rolling polyhedra. A humanoid can be approximately modeled as a 3D convex polyhedra. Given the quasi-static state of the robot, we may view the humanoid fall as a rolling polyhedra on the ground. The stepping during the tripod fall can be seen as morphing the polyhedra to reach a desired polyhedra with a new 3D convex hull. This morphing process will change the direction and the impact force of the fall.



(a) Upright standing (b) Postural balance maintenance (c) Stepping for balance maintenance (d) Tripod fall

Fig. 3. In momentum control, when both the ground reaction force and the center of pressure (CoP) computed from desired momentum rate change are not simultaneously achievable, a few distinct scenarios can arise according to the magnitude of the push. (b) Fully respecting the desired linear momentum while compromising angular momentum (if needed) results in a postural balance control without stepping. (c) Contrarily, fully respecting the desired angular momentum while compromising linear momentum (if needed) results in a stepping motion. (d) When the push is even larger than what the stepping controller can handle, compromising both the desired angular momentum and linear momentum leads to the tripod-like motion which involves a reactive stepping as well as the bending motion of the body. The control of the arm is added to complete the tripod fall controller.

A. Transition among the Fall Control Strategies using Momentum Controller

We have earlier used momentum-based controller to control both the linear and angular momenta of the robot. As the reactive stepping controller was used for a disturbance with magnitude larger than what the postural balance controller can handle (See Fig. 3), the tripod fall controller can be used for an even larger disturbance for which even a reactive stepping controller cannot prevent a fall. Therefore we use the same technique to transition from the reactive stepping control to the tripod fall control as the one for the transition between the balance control and the reactive stepping control.

We briefly review the core of the momentum based controller [6] which controls the rate changes of linear and angular momenta according to the following rules:

$$\dot{\boldsymbol{k}}_d = \boldsymbol{\Gamma}_{11}(\boldsymbol{k}_d - \boldsymbol{k}) \tag{1}$$

$$\dot{\boldsymbol{l}}_d/m = \boldsymbol{\Gamma}_{21}(\dot{\boldsymbol{r}}_{G,d} - \dot{\boldsymbol{r}}_G) + \boldsymbol{\Gamma}_{22}(\boldsymbol{r}_{G,d} - \boldsymbol{r}_G) \tag{2}$$

where $\boldsymbol{k}$ and $\dot{\boldsymbol{k}}$ are centroidal angular momentum and its rate change, m is the total mass of the robot, $\boldsymbol{r}_G$ and $\dot{\boldsymbol{r}}_G$ are the locations and the velocity of the CoM. $\dot{\boldsymbol{k}}_d$ and $\dot{\boldsymbol{l}}_d$ are the *desired* rates of change of centroidal angular and linear momenta [1], respectively, and $\boldsymbol{r}_{G,d}$ is the desired CoM position. $\boldsymbol{\Gamma}_{ij}$ represents a 3×3 diagonal matrix of feedback gain parameters. The details can be found in [6].

The key to this controller is the observation that different choices of the gain matrices yield different push recovery behaviors. For example, a small $\boldsymbol{\Gamma}_{11}$ and large values of $\boldsymbol{\Gamma}_{21}$ and $\boldsymbol{\Gamma}_{22}$ will generate the motion shown in Fig. 3(b). On the other hand, a large $\boldsymbol{\Gamma}_{11}$ causes the robot to respect angular momentum more strictly. When the desired angular momentum is zero, which is reasonable for stepping motion, the controller would move the robot CoM position rather than rotate the trunk. By adding a low-level stepping controller to this momentum controller, reactive stepping was achieved as shown in Fig. 3(c).

Next we explain how to transition from this reactive stepping controller to the tripod fall controller. As shown in Fig. 3(d), the tripod fall motion includes both stepping and bending down of the trunk of the humanoid. Thus $\boldsymbol{\Gamma}_{11}$, $\boldsymbol{\Gamma}_{21}$ and $\boldsymbol{\Gamma}_{22}$ are tuned to generate those motions. $\boldsymbol{\Gamma}_{11}$ is chosen so that it is smaller than the one for reactive stepping but greater than the one for balance maintanence, and $\boldsymbol{\Gamma}_{21}$ and $\boldsymbol{\Gamma}_{22}$ are also set in between the reactive stepping values and the balance maintanence values.

B. Control Algorithm

The main control algorithm for the tripod fall is as follows:

- After the push is completed, the humanoid takes a step at the learned stepping location while bringing the shoulder joints to the learned shoulder angles and stretching the wrist angles.
- On the completion of stepping, the humanoid locks the leg joints. The arms are also locked after they are fully stretched.

There are six control parameters: x-y stepping location on the floor (2 parameters) and two shoulder angles for each arm ($2 \times 2 = 4$ parameters).

In future, we will consider impedance control of the contacting foot and hand at the impact time so that the controller can further reduce the impact force.

C. Reinforcement Learning Algorithm

In order to find the optimal stepping location and the optimal shoulder angles, we implement a reinforcement learning algorithm using a gradient descent method [7].

The first challenge is to determine which robot states to use to train the learning algorithm. Since a humanoid robot has a large number of degrees of freedom, exploring all the states can be impractical and time consuming. For instance, the robot in our simulation has 24 dof. In addition, there are many other important quantities such as CoM, CoP, linear/angular momentum, etc. If we explore all the states and the derived quantities, a learning algorithm has to work in a 50+ dimensional space. In order to achieve an efficient learning, we use a *reference* stepping location for training so that the six control parameters are assigned at the given stepping location. In other words, when the fall controller is activated, the robot calculates the stepping location using only the robot states (no control parameters used here) and the robot performs the tripod fall controller using the control parameters which are stored at the stepping location. Finally, the control parameters at that stepping location are updated given the impact forces at the foot and the hand.

In order to use the stepping location as the reference, it has to be unique and time invariant so that the same robot states yield the same reference stepping location. We use the *generalized foot placement estimator* (GFPE) as the reference point, which was proposed in our previous work [8] for reactive stepping. The GFPE is chosen so that the CoM will stop vertically above the stepping location after the robot takes a step, modeling the humanoid as a rimless wheel. The GFPE has the following properties: 1) defined both on level and non-level grounds, 2) unique, 3) not state-dependent and 4) fixed on the ground; once computed, it does not move with subsequent robot movement. Therefore, instead of exploring the entire robot state space, the controller updates the six parameters given the GFPE which can uniquely decided from the robot states.

We train the controller by giving a variety of pushes to the robot. Since each push results in its corresponding GFPE, and the learning algorithm trains the parameters assigned at the GFPE.

The reinforcement learning algorithm works as described in Algorithm 1. The algorithm starts with reading the current parameters corresponding to the calculated GFPE, and adds Gaussian noise to the parameters,

$$\hat{\boldsymbol{w}} = \boldsymbol{w} + \boldsymbol{Z}$$
$$\boldsymbol{Z} \backsim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma}^2)$$

where $\boldsymbol{w}$ is the current vector of the control parameters and $\hat{\boldsymbol{w}}$ is the new vector with noise $\boldsymbol{Z}$ and variance $\boldsymbol{\sigma}^2$.

Algorithm 1 Policy gradient learning algorithm

1: **for** Each trial **do**
2: Start with a push with the fixed magnitude and direction
3: Obtain $g(\boldsymbol{w})$ from the previous simulation data
4: Add a Gaussian noise $\boldsymbol{Z}$ to $\boldsymbol{w}$
5: Obtain $g(\boldsymbol{w} + \boldsymbol{Z})$ after the fall control is done
6: Update $\boldsymbol{w}$ by $\delta\boldsymbol{w}$
7: **end for**

The parameters are updated by the gradient descent method:

$$\Delta\boldsymbol{w} = -\eta(g(\boldsymbol{w} + \boldsymbol{Z}) - g(\boldsymbol{w}))\boldsymbol{e}_N$$

where η is a matrix of learning rates, $g(\cdot)$ is a cost function defined as the sum of the maximum impact forces:

$$g(\boldsymbol{w}) = \alpha F_{foot} + \beta F_{hand} \text{ when}, z_{CoM} > Z_{thresh}, \quad (3)$$
$$g(\boldsymbol{w}) = \alpha F_{foot} + \beta F_{hand} + \Gamma, \text{ otherwise}, \quad (4)$$

[1] $\dot{\boldsymbol{k}}_d$ and $\dot{\boldsymbol{l}}_d$ are *not* derivatives of $\boldsymbol{k}_d$ and $\boldsymbol{l}_d$ but are desired values for rate changes of $\boldsymbol{k}_d$ and $\boldsymbol{l}_d$.

where F_{foot} and F_{hand} are the maximum impact forces to the swing foot and the landing hand, respectively, during fall. The positive constants α and β are used to set the relative importance; both are set to 1.0 for the following simulation. $\boldsymbol{e}_N$ is an eligibility vector, updated by:

$$\boldsymbol{e}_{k+1} = \boldsymbol{e}_k + \frac{\boldsymbol{Z}}{\sigma^2},$$

where k is a number of trials. To penalize the case in which the robot cannot stabilize the tripod posture and falls to the ground, the constant penalty Γ is added when the height of the CoM z_{CoM} is lower than a threshold value.

III. SIMULATION

The tripod fall controller has been implemented in our *Locomote* software package [9] and simulated on Webots [10]. The full-sized humanoid in our simulation has two legs each with 6-dof, two arms each with 4-dof of which the two wrist joints are locked during the tripod fall control. The robot is 1.3 m tall and weighs 55 kg.

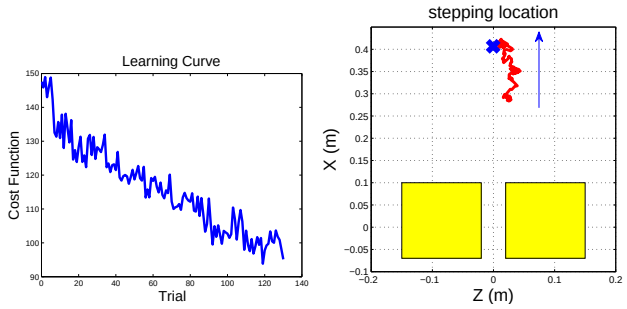
A. Training

As a training set, a number of sharp pushes are applied from behind to the center of the humanoid trunk when the humanoid is standing still with double support as shown in Fig 4. We train the learning controller for a fixed number of pushes. Each push force has two features: a magnitude and a direction. A range of 0.1sec duration push forces varying from 240N to 360N with 20N increments (a total of 7 force magnitudes), and having directions 0°, 20° and 30°, (a total of 3 directions) measured from the forward direction of the humanoid (clockwise looking from the top), were selected in the simulation. The total number of cases is therefore $7 \times 3 = 21$. The magnitudes of the pushes are chosen based on the experiences of our previous work [8] where we found out that our reactive stepping controller can survive up to the 280N for 0.1 sec push.

The initial stepping location is chosen as the mid point between the initial swing foot location and the GFPE. The initial shoulder angles are 0° and 20° respectively for each arm, which makes the arm perpendicular to the body when seen from the side. We only use the right leg as the swing leg in order to generate stable stepping motion since the pushes in the training set tended to drive the CoP to the left foot.

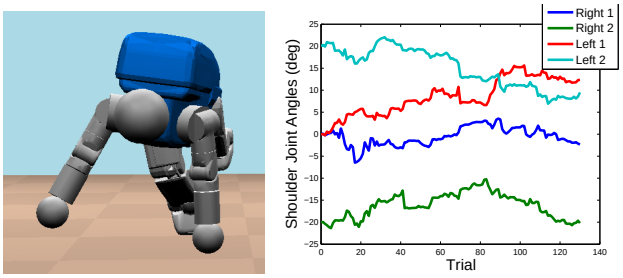
Fig. 4 shows the learned motion of the tripod controller for the (280N, 0deg) push. We trigger the controller as soon as the push ends. The robot takes a step using its right foot and lands on the ground using first the left hand and then the right hand.

The evolution of the cost function and the control parameters are shown in Figs. 5 and 6. The learning curve shows rapid convergence with some fluctuation. After 130 trials, the learning algorithm does not make any significant improvement. As shown in Fig. 5(b), the stepping location moves mostly forward which makes sense since a large step may absorb more kinetic energy by slowing down the fall. We conjecture the step location stops advancing due to the kinematic limit and the limited time available for fall.



(a) (b)

Fig. 5. Evolution of tripod fall by reinforcement learning algorithm for a push of 280N. (a) The curve shows the gradual reduction of cost function. Though the cost function does not reduce strictly monotonically, it nevertheless converges in a reasonable manner. Note the unit of the cost function is scaled force. (b) The evolving location of stepping. The two yellow boxes approximately represent the two footprints. The blue cross is the GFPE. The location moves forward close to GFPE as learning continues.



(a) (b)

Fig. 6. (a) Snapshot of the moment when the arm hits the ground. (b) Plot for the learned shoulder angles from the 280N push case. The robot touches the ground using the left hand first, and the learned angles make the left arm almost perpendicular to the ground, which makes sense since that right angle minimizes the time between stepping and the hand contact. The shoulder angles of the right arm do not contribute to the cost function since the maximum impact force of the arms always comes from the left hand, therefore the learned angles of the right arm do not converge.

The evolution of the shoulder angles in Fig. 6(b) is also intuitively understandable. Considering that the left hand hits the ground first and is likely to have the maximum impact force, the learned shoulder angles of the left arm makes the arm hit the ground in almost perpendicular as shown in Fig. 6(a), which implies that the left hand hits the ground as soon as possible in order to minimize the landing speed of the robot at the instant ground contact. The graphs of the right arm shoulder angles do not show clear tendency since landing of the right arm is unlikely to yield the maximum impact force.

Fig. 7 shows the reduction of the maximum impact force for the (280N, 0deg) push. Without fall control, the maximum impact force is about 12 kN to the knee and the head as shown in Fig 8(a). Using the tripod fall controller with the initial control parameters significantly reduces the maximum impact almost by half, and it is further diminished to 4 kN after learning. The exerted forces to robot bodies in Fig 8 show that without fall control three different bodies (knee, head and trunk) have larger impact forces than the maximum impact force from the tripod fall control.

Fig. 9 shows the time evolution plots of potential and

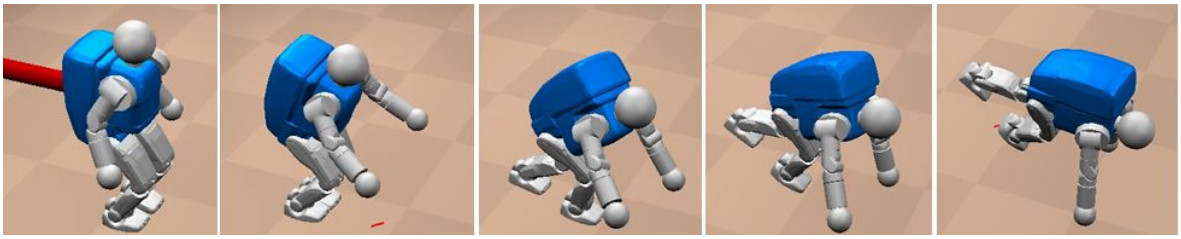


Fig. 4. Snapshots showing a sequence of robot configurations of the learned tripod fall control of a full-sized humanoid having two 6-dof legs and two 4-dof arms. The two wrist and the two elbow joints are locked so that the arms are fully stretched. The red cylinder in the very left snapshot indicates a push of 280N with a duration of 0.1 sec. The robot takes a step and lands on the ground by the two hands.

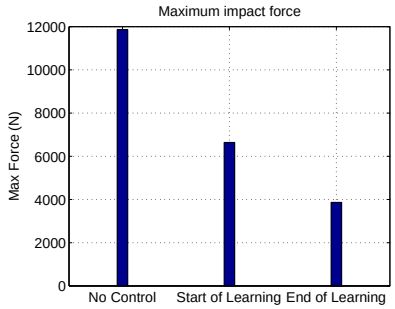


Fig. 7. The maximum impact force during fall from the 280N push case. Without fall control, either knee or head receives the maximum impact force during fall. For the 280N case, the maximum impact is occurred at the knee contact as shown in Fig 8. Even with the unlearned controller, the maximum impact force is reduced about by almost half, and a trained controller shows three times smaller maximum impact force to the arm.

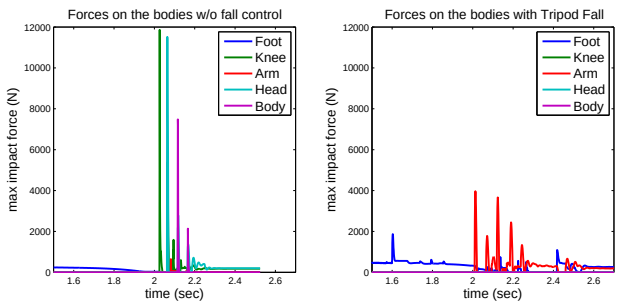


Fig. 8. Exerted forces on the robot bodies during fall from the 280N push case. Without fall control, the knee and the head have the greatest impact force at the first and second contact to the ground. The tripod controller significantly reduces the peak force which occurs at the hands.

kinetic energies during humanoid fall. Both plots start at the same energy with large PE and small KE. Without fall control, most of the PE is converted to the KE which contributes to the higher speed at impact. In contrast, the big gap between the PE curve and the KE curve proves that the tripod fall controller conserves a large portion of the PE even after fall and consequently the velocity at the impact is much smaller. The PE versus KE plot in Fig. 10 highlights that difference. The max KE of the no fall control case is about 147 J and that of the tripod fall control is about 40 J.

All the results from the training set are summarized in Table I. In every training item, the tripod fall controller makes significant reduction of the maximum impact force.

Fig. 11 shows the trained stepping locations according to

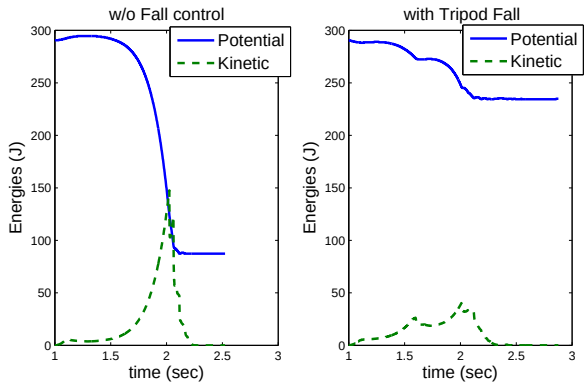


Fig. 9. Time-energy plots from the 280N push case. The height of the CoM is used for the potential energy and the velocity of the CoM is for the kinetic energy for simplicity. The plots show the significant reduction of energy conversion by the tripod fall controller. The first bump of the right plot comes from stepping and the two following bumps denotes moments of the hand-landing.

the GFPEs. Though the GFPEs (blue crosses) clearly display the angles of the pushes, the learned stepping locations (red circles) are more biased to the right. We conjecture that this bias comes from the fact that the robot must rotate around the leading edge of the convex hull as we discussed earlier when we model a humanoid as a polyhedra. Even though the instantaneous CoM velocity after the push shows the direction of the push (the GFPE uses the CoM velocity for its computation), the left directional portion of the CoM velocity diminishes as the robot eventually falls around the front edge of the two feet or only the left foot.

IV. EXPERIMENTS

Since the tripod fall experiments are likely to damage a full-sized humanoid robot, we have performed our experiments of fall using the smaller Aldebaran NAO H25 robot [9]. Due to the limitation in the hardware control interface and force sensing capability, we play-back the pre-recorded joint angle trajectories from simulations on the robot hardware instead of running the online reinforcement algorithm.

In order to generate the joint angle trajectories, we repeat the online reinforcement learning algorithm for the NAO robot in simulation. Since it is very hard to generate a precise impulse force in experiments, a steady and slow push is given to the NAO robot in simulation until the lean angle of the

Push		Maximum impact force (N)		
Magnitude (N)	Angle (deg)	No control	Start of Learning	End of Learning
240	0	11800	5400	3200
	20	11100	5600	3700
	30	10100	6000	3700
260	0	12100	5100	2300
	20	11700	5400	3000
	30	11700	5400	1500
280	0	11900	6600	3900
	20	11700	6300	4000
	30	12000	6700	3900
300	0	11700	5700	4000
	20	11000	5800	4300
	30	10100	6100	3900
320	0	11100	5100	3600
	20	12100	5400	3900
	30	11000	5700	4300
340	0	12200	5000	3400
	20	11700	5700	3900
	30	10700	6200	3700
360	0	11800	4100	3300
	20	12200	5000	4100
	30	10600	6700	3800

TABLE I

REDUCTION OF THE MAXIMUM IMPACT FORCE BY REINFORCEMENT LEARNING ALGORITHM. THE PUSH MAGNITUDES RANGE FROM 240N TO 360N IN 20N STEPS; EACH PUSH LASTS FOR 0.1 SEC, AND IS APPLIED ON THE ROBOT HORIZONTALLY MAKING THREE ANGLES FROM THE FRONT DIRECTION: 0, 20, 30 DEGREES. THE MAXIMUM IMPACT FORCES ARE ROUNDED AT 100N.

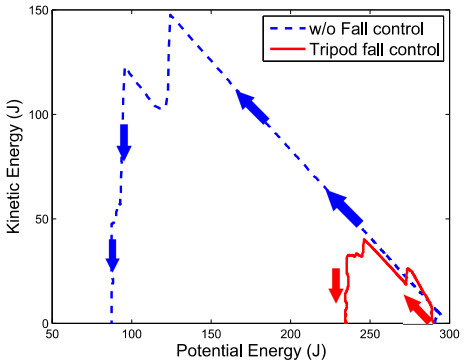


Fig. 10. Potential energy (PE) versus kinetic energy (KE) plot from the 280N push case. The curves start from the lower right corner with the largest PE and zero KE. The arrows indicate the direction time evolution on the plots. The plots show that the robot falling under tripod control strategy retains a much larger PE after the fall compared with the no-fall-control case.

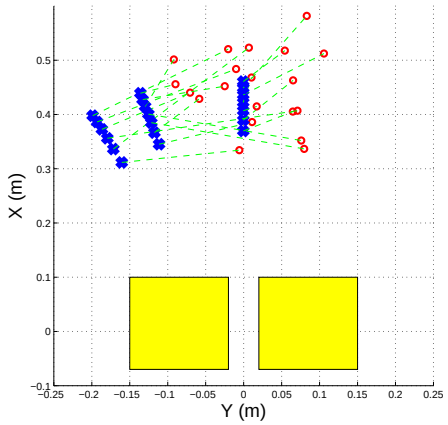


Fig. 11. Trained stepping locations. The blue crosses are the GFPEs corresponding to the pushes, and the red circles are the learned stepping locations connected to the GFPEs by the dotted green lines. The yellow boxes are the initial footprints.

robot exceeds the pre-determined 15 degrees threshold, at which the robot is considered to start falling.[2]

Given the pre-recorded trajectories, the robot is controlled to follow the trajectories using the built-in function for the trajectory control of the NAO. The controller runs on an external laptop connected to the robot via a wired network. The lean angle of the robot is estimated from the IMU. More detailed setup of the experiment can be found in [9].

Since the NAO robot does not have force sensors in the hand or wrist, we measure the impact force at the hands indirectly using the acceleration data from the IMU. Fig. 13 shows the accelerations from the two experiments. In each case we see a significant reduction of the peak acceleration.

[2]This threshold was also used in our earlier fall experiments [9].

V. CONCLUSIONS AND FUTURE WORK

In order to reduce the damage to a humanoid robot caused by a fall, we proposed a new strategy called the tripod fall control. In this approach the robot adopts a tripod like posture during the fall by deliberately using three consecutive contacts, two hands and one foot. The strategy ensures that only a fraction of the robot's potential energy is converted into kinetic energy so that the resultant velocity at the impact, and consequently the extent of damage, is reduced.

To simplify the control problem, the strategy controlled only six parameters: step location (x and y) and shoulder joint angles (2 per arm). Due to the highly non-linear dynamics of the humanoid, we used reinforcement learning

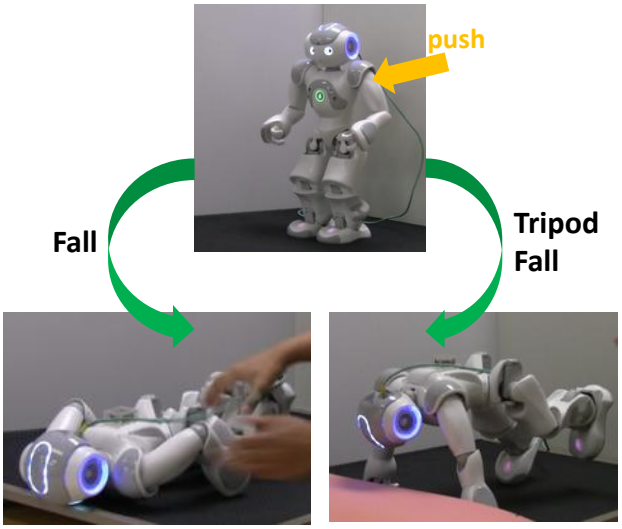


Fig. 12. Consequence of a humanoid fall without and with the proposed tripod fall controller. The Aldebaran NAO robot is subjected to a push from behind (top figure) in upright stance pose. Without any fall controller the robot falls to the ground where the entire potential energy is transformed to kinetic energy resulting in high impact velocity which may damage the robot. The tripod fall controller uses the right foot and two hands to keep the CoM high so that the velocity at impact is lowered and damage is reduced.

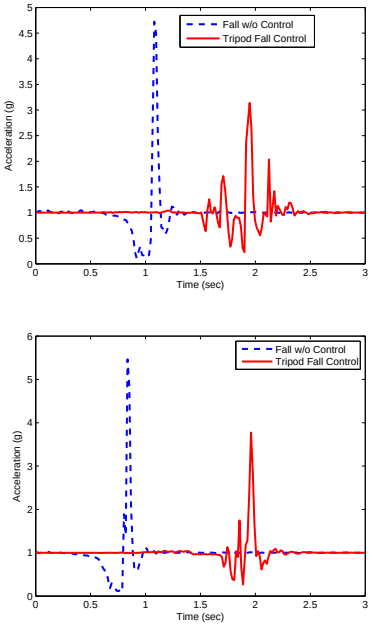


Fig. 13. The acceleration data of the trunk from the IMU during the falls of the NAO robot. The dotted blue curve shows the acceleration without any fall control. The default reactive fall controller was turned off. The solid red curve is data from the tripod fall controller. The two experiments are executed with the identical initial conditions.

algorithm to search for the optimal values of these parameters. For efficient training, the entire state of the robot was projected on the generalized foot placement estimator (GFPE) [8]. The control parameters were assigned at the GFPE of each training case as if the GFPE is an index for the parameters.

The simulation results showed that our strategy results in a significant reduction of the maximum impact force. The resultant motions suggested that the robot should take a long step and the arms should make contact with the ground in a perpendicular configuration, which matched our intuition.

Our controller was experimentally implemented on a NAO robot in order to prove our concept. Due to the limited sensing capability of the robot, we measured the downward trunk acceleration rather than the contact force.

We can identify the following extensions to this work. Though the simulation and experimental results showed significant reduction of the maximum impact force using our strategy, the remaining impact force is still large enough to cause damage to the robot's hands and arms. In future, more sophisticated controllers such as an impedance control for the arm contact or attaching soft material on the possible contact areas of the robot can be useful

Additionally, one can perform hardware experiments of reinforcement learning using additional force sensors at the robot feet and hands.

A more general approach should include other 3D fall scenarios such as backward falls and falls involving shoulder contact. Finally, it will be instructive to compare the presented fall strategy to the reactive strategy during an accidental human fall.

VI. ACKNOWLEDGMENTS

We wish to thank the anonymous reviewers for valuable comments.

REFERENCES

[1] K. Fujiwara, F. Kanehiro, S. Kajita, and H. Hirukawa, "Safe knee landing of a human-size humanoid robot while falling forward," in *IEEE/RSJ Intn'l Conf. on Intelligent Robots and Systems (IROS)*, September 28– October 2 2004, Sendai, Japan, pp. 503–508.

[2] 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 *Humanoids06*, 2006, pp. 524–529.

[3] K. Ogata, K. Terada, and Y. Kuniyoshi, "Falling motion control for humanoid robots while walking," in *Humanoids07*, Pittsburgh, 2007.

[4] ——, "Real-time selection and generation of fall damage reduction actions for humanoid robots," in *Humanoids08*, Dec. -3 2008, Daejeon, Korea, pp. 233–238.

[5] 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 Intn'l Conf. on Intelligent Robots and Systems (IROS)*, Sep 2002, Lausanne, Switzerland, pp. 2521–2526.

[6] S.-H. Lee and A. Goswami, "Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground," in *IEEE/RSJ Intn'l Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, Oct 2010, pp. 3157–3162.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[8] S. Yun and A. Goswami, "Momentum-based reactive stepping controller on level and non-level ground for humanoid robot push recovery," in *IEEE/RSJ Intn'l Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, USA, Sep 2011.

[9] ——, "Humanoid robot safe fall using aldebaran NAO," in *IEEE Intn'l Conf. on Robotics and Automation (ICRA)*, St. Paul, USA, Apr 2012.

[10] O. Michel, "Webots: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.