2011 IEEE International Conference on Robotics and Automation
Shanghai International Conference Center
May 9-13, 2011, Shanghai, China

Learning Full Body Push Recovery Control for Small Humanoid Robots

Seung-Joon Yi[*][†], Byoung-Tak Zhang[†], Dennis Hong[‡] and Daniel D. Lee[*]

Abstract— **Dynamic bipedal walking is susceptible to external disturbances and surface irregularities, requiring robust feedback control to remain stable. In this work, we present a practical hierarchical push recovery strategy that can be readily implemented on a wide range of humanoid robots. Our method consists of low level controllers that perform simple, biomechanically motivated push recovery actions and a high level controller that combines the low level controllers according to proprioceptive and inertial sensory signals and the current robot state. Reinforcement learning is used to optimize the parameters of the controllers in order to maximize the stability of the robot over a broad range of external disturbances. The controllers are learned on a physical simulation and implemented on the Darwin-HP humanoid robot platform, and the resulting experiments demonstrate effective full body push recovery behaviors during dynamic walking.**

Keywords: Full Body Push Recovery, Reinforcement Learning, Humanoid Robots

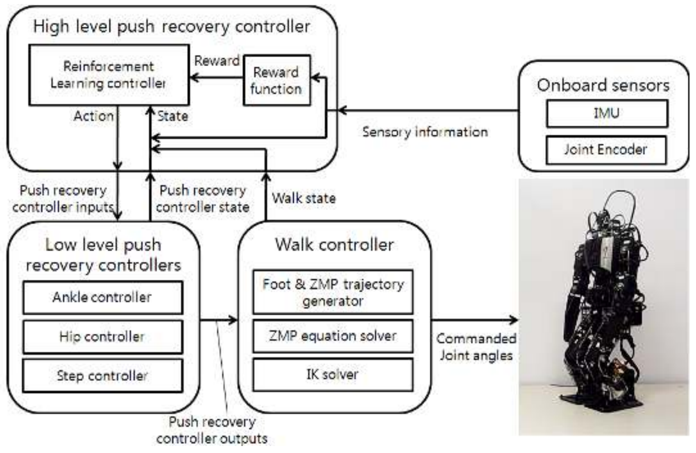

High level push recovery controller
Reinforcement Learning controller
Reward
Reward function
Action
State
Sensory information
Onboard sensors
IMU
Joint Encoder
Push recovery controller inputs
Push recovery controller state
Walk state
Low level push recovery controllers
Ankle controller
Hip controller
Step controller
Walk controller
Foot & ZMP trajectory generator
ZMP equation solver
IK solver
Commanded Joint angles
Push recovery controller outputs

Fig. 1. Overview of the hierarchical push recovery controller.

I. INTRODUCTION

Humanoid robots are not intrinsically stable, so stumble and push recovery are critical to allow them to operate in unconstrained environments. For example, typical living and office environments are full of potential objects and people the robot could collide with, and the walking surfaces contain surface irregularities such as bumps and debris the robot could step upon. Even if the environment is perfectly flat and clear of obstacles, simple open loop walk controllers fail due to imperfect modeling of the robot and its actuators.

Thus, the problem of push recovery has been a topic of major interest in humanoid robot research. One approach in this area uses a full dynamic model of the robot and very fast feedback control to reject external forces applied to the robot using force controlled actuators [1], [2]. However, such an approach is quite difficult to implement on currently existing humanoid platforms as they require multi-axis force sensors and force controlled actuators, in addition to a precise dynamic model of the robot and a large amount of computational processing power.

Another direction in push recovery research focuses on biomechanically motivated, simple push recovery behaviors humans are known to perform in response to unexpected perturbations [3], [4], [5], [6], [7], [8]. In this approach, a simplified model of the robot is used to analytically derive appropriate recovery responses given knowledge of the robot state. Although this approach is simple in principle, there are a number of practical issues before successful implementation on generic humanoid robot platforms. In particular, these methods assume precise knowledge of the dynamical robot state as well as force controlled actuators. Most of these methods have also been only demonstrated using a single push recovery controller on a stationary robot. So there are still many open questions concerning how to construct a robust and generic push recovery behavior for humanoid robots similar to what humans display.

The aim of our work is to design a practical, situationally-aware full body push recovery strategy for humanoid robots that uses only

- commonly available sensory data
- position controlled actuators
- relatively slow feedback control

To fulfill such requirements, we employ a hierarchical control structure shown in Figure 1. We use three low-level biomechanically motivated push recovery controllers, whose outputs are fed into a walk controller that generates joint position commands for all the actuators. A high level controller receives state information from proprioceptive and inertial sensors as well as all the low level controllers, and generates high level command parameters for the lower-level controllers based upon this information. However, this approach has two main problems that need to be resolved. We cannot directly employ an analytically derived push recovery control as the precise physical state of the robot such as

[*] GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104 {yiseung,ddlee}@seas.upenn.edu
[†] BI Laboratory, Seoul National University, Seoul, Korea. btzhang@ceas.snu.ac.kr
[‡] RoMeLa Laboratory, Virginia Tech, Blacksburg, VA 24061 dhong@vt.edu

the center of mass velocity is not available from the simple sensors, and it is difficult to design the high level controller without proper analytic model.

We solve both design problems by learning an appropriate controller using training examples [9]. We use a reinforcement learning (RL) algorithm which finds the optimal mapping from a simplified state space to recovery actions that maximizes a predetermined reward function over time. Controllers are trained using repeated trials to maximize the stability of the robot under external disturbances. Another advantage of this approach is that as it can use a full-body humanoid model or even the physical robot to learn the controller, it has potential to learn better controller than analytic approaches assuming simplified models.

This paper is organized as follows. Section II introduces the three low level push recovery actions and explains how we implement them on a humanoid robot with position controlled actuators. Section III explains the high level controller that modulates the low level push recovery actions, and how the machine learning optimization problem is defined. Section IV addresses the learning of the full push recovery controller from experience in a simulated environment, and Section V shows the experimental results when the trained push recovery controller is implemented on a small humanoid robot. Finally, we conclude with some potential issues and future directions arising from this work.

II. Biomechanically Motivated Push Recovery Controllers for Humanoid Robots

Biomechanical studies show that human beings display three distinctive motion patterns in response to a sudden external perturbation, which we will denote as ankle, hip and step recovery strategies. In this section, we will introduce each of the three push recovery actions in detail, and explain how these push recovery strategies can be implemented on a humanoid robot with position-controlled actuators.

A. Ankle Controller

The ankle controller is a push recovery strategy which keeps the center of mass (CoM) within the base of support (BoS) by applying control torque on ankle joints. It can be implemented by using a simple p-control on ankle torque, but it cannot be directly implemented on generic robots without torque controlled actuators. One option is controlling the target position of ankle actuators [10]. However we have found this approach has some practical limitations as the feet of small humanoid robots tend to get tipped under disturbance, rendering the direct control of ankle less effective.

Instead, we take the indirect approach of controlling the zero moment point (ZMP) by adding an auxiliary zero moment point (ZMP) p^{aux} that can be controlled in real time to augment the reference ZMP trajectory p^{ref} as in [11], [12].

$$ZMP_{target} = p^{ref} + p^{aux} \tag{1}$$

If we assume the linear inverse pendulum model (LIPM) and walk controller that controls the movement of CoM so

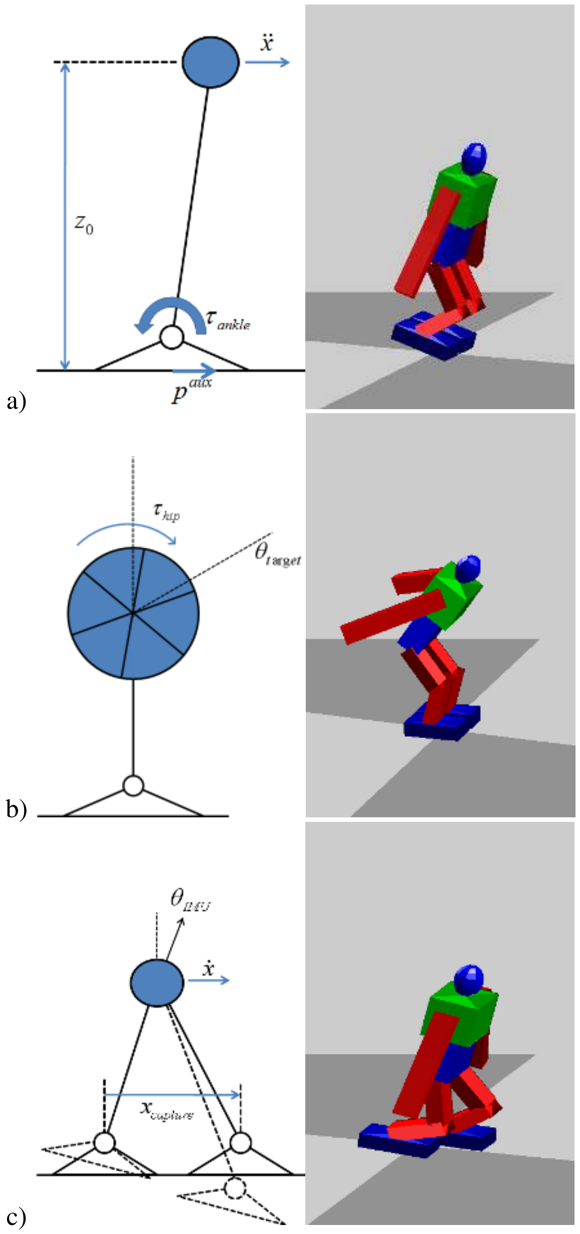


Fig. 2. Three biomechanically motivated push recovery controllers for a position controlled humanoid robot: a) ankle controller, b) hip controller, c) step controller

that the actual ZMP of the robot follows the target ZMP trajectory, the effective inertia force applied on the torso is

$$f = -\frac{Mg}{z_0} p^{aux}, \tag{2}$$

where M is the mass of torso, g the gravity constant, and z_0 is the height of CoM. This corresponds to an effective ankle torque of

$$\tau_{ankle} = Mgp^{aux} \tag{3}$$

In this manner, we can indirectly implement the ankle recovery strategy by controlling the auxiliary ZMP p^{aux}, given a ZMP based walk controller.

B. Hip Controller

The hip controller is a push recovery strategy which uses angular acceleration of the torso and limbs to counter the movement of the CoM. When a human is pushed from behind, he rotates his torso towards the front. This seemingly counterintuitive motion generates backward ground reaction force, which pulls the CoM back towards the BoS.

An analytic analysis uses the flywheel model in [5]. We employ a simple torque profile in the form of bang-bang control on the hip:

$$\tau_{hip}(t) = \tau_{max}u(t) - 2\tau_{max}u(t - T_{R1}) + \tau_{max}u(t - T_{R2}) \quad (4)$$

$$\theta(T_{R2}) = \theta_{max} \quad (5)$$

where τ_{max} is the maximum torque that the joint can apply, $u(t)$ is the unit step function, T_{R1} is the time the torso stops accelerating, T_{R2} is the time torso comes to a stop and θ_{max} is the maximum joint angle. We can further control the influence on angular momentum by controlling T_{R1} and T_{R2}.

We implement this controller with position controlled actuators by specifying the target angle of the hip actuator as a control input θ_{target}. A position controlled actuator with high proportional gain applies maximum torque τ_{max} until the hip angle almost reaches θ_{target}, and applies negative torque to stop, which approximates the bang-bang control profile shown above. After reaching the target position, the controller should move the hip angle back to its initial value. We add a binary state variable PS_{hip} to implement this two-phase behavior.

C. Step Controller

The step controller is a push recovery strategy which effectively moves the BoS by taking a step. This strategy has been studied using the capture point concept, defined as the point on the ground where the robot can step in order to bring itself to a complete stop [5]. Analytical analysis with the flywheel model shows that the theoretical capture point of a robot without torso momentum control is proportional to the velocity of torso

$$x_{capture} = \dot{x}\sqrt{\frac{z_0}{g}}. \quad (6)$$

We implement this controller by extending the walk controller so that the target foot position can be overridden with the control input $x_{capture}$, so that the next step goes to desired capture point. A binary state variable PS_{step} is used to denote whether the walk controller is overridden or not.

However, we have found there are some implementation issues with such a modification of the walk controller. The robot may easily become tipped, which makes the foot kick the ground during stepping and lose balance. The support foot also needs to be appropriately chosen so that it keeps contacting the ground during this perturbation. We have made three modifications to handle the special case of stepping under large perturbation. First, the optimal swing leg is determined based on the current configuration of feet, the phase of walking and the direction of perturbation so that the current pivot foot should not be chosen as the stepping foot. Second, the height and orientation of both feet are modified using inertial sensors during single support phase so that the swing foot lands squarely on the surface. Finally we apply lower the electrical motor compliance of the swing leg to reduce the shock of abrupt landing.

III. High Level Push Recovery Controller

In the previous section, we explained three biomechanically motivated push recovery strategies and how these push recovery controllers are implemented on a humanoid robot with position controlled actuators.

But human beings are also capable of performing more generic, situationally aware push recovery behavior based upon composing those simple push recovery strategies. For example, humans rely on the ankle strategy when the perturbation is small, and then use a combination of strategies when the perturbation becomes larger. Humans rely more on the ankle strategy when their posture is stable, and use other strategies in more unstable positions.

To be able to mimic such push recovery behaviors, we use a high level controller that determines when and how to use the low level push recovery controllers based on current sensory information and the current state of the low-level controllers. We do not have a good analytical model for such a task, and hand designing such a controller without analytical model can be very difficult. Instead, we rely upon a machine learning approach to learn an appropriate controller from experience.

In this section, we briefly explain the walk controller we use, and the walk state and sensory information which comprise the inputs of the high level controller. Finally we describe the high level controller in detail, and explain how reinforcement learning is used to determine the appropriate parameters.

A. Walk Controller and Walk State

We use a periodic, linear inverse pendulum model (LIPM) based walk controller. As the target application of our robot requires agile omindirectional movement, ZMP preview control [13] is not utilized due to its control latency. Instead, we assume a piecewise linear ZMP trajectory with endpoint constraints, and use an analytical solution of the ZMP equation to calculate the torso trajectory. Foot and ZMP trajectories are generated in real time using the current state and command velocities, and the torso position is calculated to satisfy the ZMP equation. Although the walk controller itself has a number of discrete and continuous states, we use a single discrete walk state as input to the push recovery controller:

$$WS = \{STOP, DS_1, SS, DS_2\} \quad (7)$$

where each state is defined as follows:

- $STOP$: double support state where CoM and ZMP of the robot lie on the middle point of two feet.
- DS_1: double support state where ZMP and COM is moving towards current support foot.

- *SS*: single support state where ZMP lies in the supporting foot.
- *DS_2*: double support state where ZMP and CoM is moving back to the middle point of two feet.

B. Sensory Information

One of our aim is to employ commonly available sensors for sensory feedback. One popular option is an inertial measurement unit (IMU), which consists of inexpensive rate gyroscopes and accelerometers mounted on the torso of the robot. Typically these signals are filtered to get an estimate of the torso angle, which is used to determine the current posture of the robot [14]. We use the filtered torso angle, θ_{IMU}, as the first sensory feedback signal.

In addition to the filtered torso angle, raw gyroscope and accelerometer signals have also been used for push recovery feedback, but gyroscope are more commonly used since they are not corrupted by linear motion of the robot [10]. In this work, we also use the raw gyro value θ_{gyro} as an additional sensory feedback signal.

Finally, we use the current joint encoder angles from the actuators. With this information and the forward kinematics model of the robot, we can calculate the relative orientation of the feet from the torso. With the torso angle estimate, we can determine how the current support foot is tipped. We use this estimate of foot tip angle θ_{foot} as the final sensory feedback signal.

C. High Level Push Recovery Controller

The high level controller should choose appropriate parameters for the three low level controllers based upon the current walk state and sensory information. It is difficult to derive an analytic controller without full knowledge of the current state, and designing such a controller by hand is not very practical. Instead, we formulate the problem as a reinforcement learning (RL) problem and train it using past experience; such an approach has been previously demonstrated in other robots [10], [15], [16], [17], [18].

RL learns the policy π, which is a mapping from state S to action A, in order to maximize long term reward R. We can formalize the high level control task as a RL problem by defining the key elements of RL: S, A and R. The state S can be defined as a combination of current sensory information and the walk and push recovery controller states

$$S = \left\{\theta_{IMU}, \theta_{gyro}, \theta_{foot}, WS, PS_{hip}, PS_{step}\right\} \tag{8}$$

The desired action A are the inputs to the three low level push recovery controllers

$$A = \left\{p^{aux}, \theta_{target}, x_{capture}\right\} \tag{9}$$

Finally, we use a reward R defined as

$$R = \left|\theta_{gyro}\right|^2 + \frac{g}{z_0}\left|\theta_{IMU}\right|^2 \tag{10}$$

which is similar to the residual stopping energy used in [9]. As we formalize the controlling task as a RL problem, we can use well-known RL algorithms to train the controller.

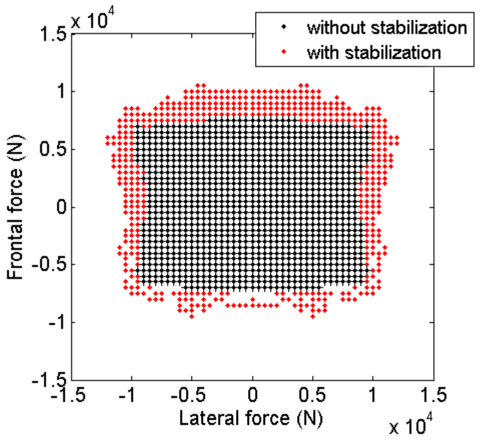


Fig. 3. Comparison of the resulting stability region over a range of impulse perturbations.

However its continuous, high dimensional state and action space makes it difficult to efficiently learn the appropriate parameters using limited training data.

Thus, we adopt three simplifications over the general formalization shown above: a) using a parameterized policy function, b) factoring the state and action spaces and c) discretizing the state space. In this simplified formulation, the low level controller inputs are defined as a parameterized function of continuous sensory information f with adjustable parameter w_i.

$$p^{aux} = f(\theta_{IMU}, \theta_{gyro}, \theta_{foot}, w_1) \tag{11}$$

$$\theta_{target} = f(\theta_{IMU}, \theta_{gyro}, \theta_{foot}, w_2) \tag{12}$$

$$x_{capture} = f(\theta_{IMU}, \theta_{gyro}, \theta_{foot}, w_3) \tag{13}$$

We discretize the state by quantizing the sensory information $\hat{\theta_{IMU}}, \hat{\theta_{gyro}}, \hat{\theta_{foot}}$

$$\hat{S} = \left\{\hat{\theta_{IMU}}, \hat{\theta_{gyro}}, \hat{\theta_{foot}}, WS, PS_{hip}, PS_{step}\right\} \tag{14}$$

and actions are now the parameter sets for each low level controller

$$\hat{A} = (w_1, w_2, w_3) \tag{15}$$

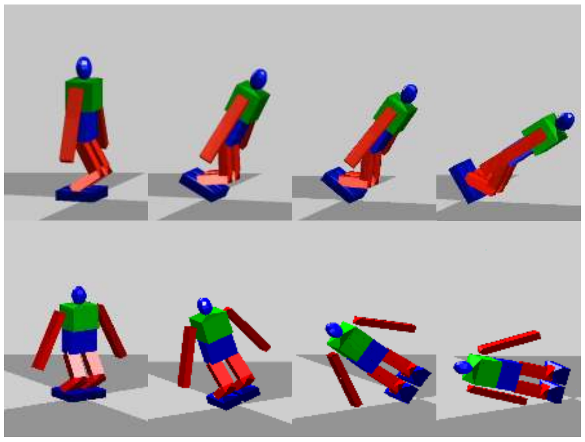
which may also be quantized to simplify the learning problem. The reward function R is unchanged from before.

IV. Learning Push Recovery Controller

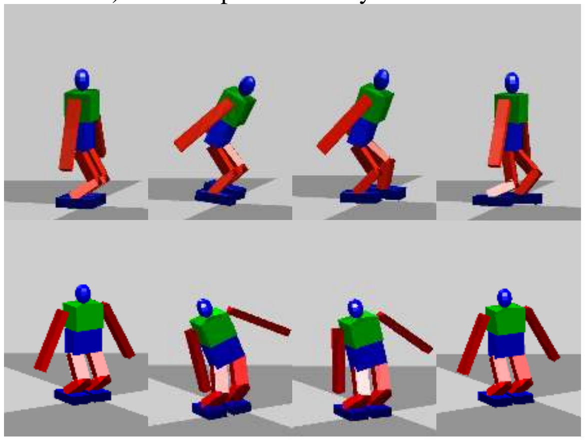
We ran a number of trials in physical simulation to train the push recovery controller. In this section we discuss the details of training the push recovery controller.

A. Simulation Setup

We have built a open-source simulation environment by integrating the Open Dynamics Engine (ODE) with Matlab based controllers and graphics, which provides us a great controllability and repeatability needed for machine learning setup. The robot model is modeled after the actual physical properties of the DARwIn-HP robot, and each joint is controlled by a high gain p-controller. The controller update frequency is set to 50Hz, and a time step of 0.0001s is used for the physics simulation.

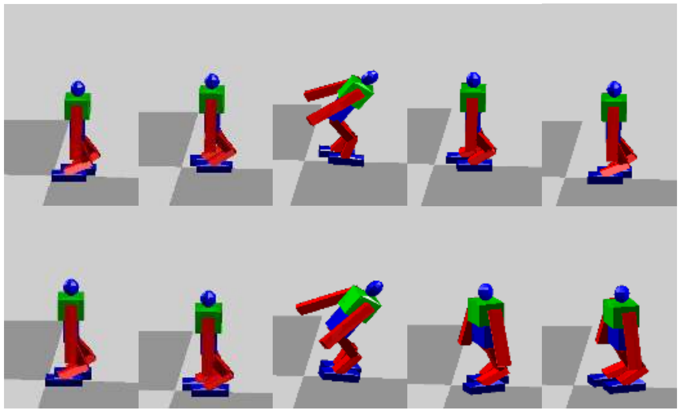


a) without push recovery controller

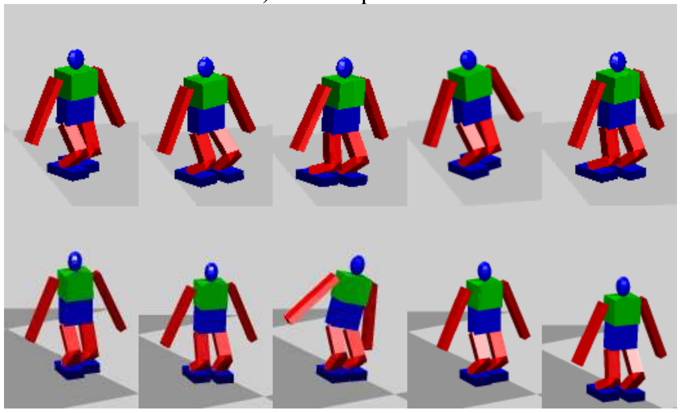


b) with push recovery controller

Fig. 4. Learned push recovery controller in the simulated environment. The same amount of frontal and lateral impulse force is applied at the torso of the robot.



a) frontal push



b) lateral push

Fig. 5. Full body push recovery during walking. The robot is set to walk forward at 12 cm/s. The same direction and magnitude of impulse force is applied for both a) and b). We see the robot reacts differently to the same perturbation according to the current walk state.

B. Reinforcement Learning Setup

To train the controller, we use the stochastic policy gradient RL algorithm [10], [17], [18] which randomly generates a number of test policies around the current best policy and use stochastic gradient descent to improve the policy. For the parameterized policy function f, we use a linear function over all inputs $\theta_{IMU}, \theta_{gyro}, \theta_{foot}$ with separate deadband, gain, punch and saturation values as parameters w_i. At each trial, the robot is reset to an initial standing pose and an impulsive force is applied for one time step. Each trial lasts for 2 seconds and the reward values are averaged over the trial period to evaluate the policy. 20 trials are done at each episode.

C. Results

Figure 3 shows stability of the robot under a broad range of two dimensional perturbations after 100 episodes of training. We see that the learned push recovery controller can balance the robot under a wider range of perturbations. Figure 4 compares the outcome of frontal and lateral perturbation with and without the learned push recovery controller. Again we can see that our push recovery controller can perform a realistic combination of push recovery strategies, and helps the robot withstand sudden perturbation that would otherwise cause it to fall down. Figure 5 shows the push recovery behavior while the robot is walking forward. We see that our push recovery controller performs the appropriate recovery action depending upon the current state of it walking.

V. Experimental Results

After training the push recovery controller in the simulated environment, we tested our algorithm using a small humanoid robot. In this section we briefly introduce the DARwIn-HP robot platform, and the experimental evaluation of the push recovery controller using this experimental platform.

A. Experimental Setup

We use the humanoid robot DARwIn-HP made by the RoMeLa laboratory at Virginia Tech. It is 56 cm tall, weighs 4 kg, and has 20 degrees of freedom. It has a web camera for visual feedback, and a 3-axis accelerometer and 2-axis gyro for inertial sensing. Position controlled Dynamixel servos are used for actuators, which are controlled by a microcontroller connected to an Intel Atom based embedded PC at a control frequency of 60hz. In this work, we use the controller trained using the simulated environment without additional training.

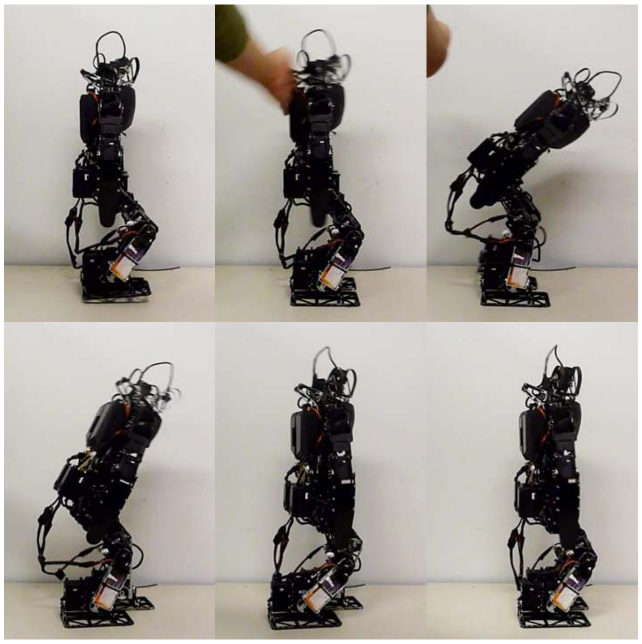


Fig. 6. Push recovery behavior during walking by a DARwIn-HP robot. The robot was set to walk in place and then pushed from behind.

We do not use arm motions for the hip recovery strategy on the physical robot, as repeated falls during the experimental trials tend to break the arm servos if powered on.

B. Results

Figure 6 shows the push recovery behavior the robot displays during walking[1]. We can see that the push recovery trained using the simulator works without modification on the physical robot. It can also be clearly seen that the robot can withstand larger perturbations with the push recovery controllers than without them during walking.

VI. Conclusions

We have proposed a practical method to implement a full body push recovery controller on a general humanoid robot without specialized sensors and actuators. Three types of biomechanically motivated push recovery behaviors are implemented by low level controllers, which are modulated by a high level controller based on inertial and proprioceptive sensory inputs. The hierarchical controller is trained using reinforcement learning to improve the push recovery performance, and the learned parameters can be used in both the simulation environment as well as on a physical humanoid robot.

Our approach has a number of advantages over previous approaches. As it is based on a number of well-motivated simpler push recovery actions, it does not require high processing power or specialized hardware with triaxial force sensors and torque controlled actuators. Yet it is capable of performing a combination of push recovery strategies based upon the current state of the robot, without a precise dynamical model of the environment.

[1]`http://www.youtube.com/watch?v=fhTa4wTUN-o`

Future work includes incorporating more efficient learning algorithms, using perception to anticipate uneven terrains, and applying these recovery strategies to more complex dynamical motions.

Acknowledgments

We acknowledge the support of the NSF PIRE program under contract OISE-0730206. This work was also partially supported by NSF grant (CNS-0958406), IT R&D program of MKE/KEIT (KI002138, MARS), NRF grant of MEST (314-2008-1-D00377, Xtran), and the BK21-IT program of MEST.

References

[1] S.-H. Hyon and G. Cheng, “Disturbance rejection for biped humanoids,” *IEEE International Conference on Robotics and Automation*, pp. 2668 –2675, apr. 2007.

[2] J. Park, J. Haan, and F. C. Park, “Convex optimization algorithms for active balancing of humanoid robots,” *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 817–822, 2007.

[3] B. Jalgha and D. C. Asmar, “A simple momentum controller for humanoid push recovery,” in *the FIRA RoboWorld Congress 2009 on Advances in Robotics*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 95–102.

[4] M. Abdallah and A. Goswami, “A biomechanically motivated two-phase strategy for biped upright balance control,” in *In International Conference on Robotics and Automation*, 2005, pp. 2008–2013.

[5] J. Pratt, J. Carff, and S. Drakunov, “Capture point: A step toward humanoid push recovery,” in *in 6th IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 200–207.

[6] T. Komura, H. Leung, S. Kudoh, and J. Kuffner, “A feedback controller for biped humanoids that can counteract large perturbations during gait,” in *Proc. IEEE Int. Conf. Robotics and Automation*, 2005, pp. 1989–1995.

[7] D. N. Nenchev and A. Nishio, “Ankle and hip strategies for balance recovery of a biped subjected to an impact,” *Robotica*, vol. 26, no. 5, pp. 643–653, 2008.

[8] B. Stephens, “Humanoid push recovery,” in *Proceedings of the IEEE RAS International Conference on Humanoid Robots*, 2007.

[9] J. Rebula, F. Canas, J. Pratt, and A. Goswami, “Learning capture points for bipedal push recovery,” in *Proc. IEEE Int. Conf. Robotics and Automation*, may. 2008, pp. 1774 –1774.

[10] F. Faber and S. Behnke, “Stochastic optimization of bipedal walking using gyro feedback and phase resetting,” in *in 7th IEEE-RAS International Conference on Humanoid Robots*, 2007.

[11] S. Kajita, M. Morisawa, K. Harada, K. Kaneko, F. Kanehiro, K. Fujiwara, and H. Hirukawa, “Biped walking pattern generator allowing auxiliary zmp control,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, oct. 2006, pp. 2993 –2999.

[12] T. Takenaka, “The control system for the honda humanoid robot,” *Age Ageing*, vol. 35, no. suppl_2, pp. ii24–26, September 2006.

[13] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, and K. Yokoi, “Biped walking pattern generation by using preview control of zero-moment point,” in *IEEE International Conference on Robotics and Automation*, 2003, pp. 1620–1626.

[14] R. Renner and S. Behnke, “Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, oct. 2006, pp. 2967 –2973.

[15] J. Morimoto, J. Nakanishi, G. Endo, G. Cheng, and P. Map, “Poincare-map-based reinforcement learning for biped walking,” in *Proc. IEEE Int. Conf. Robotics and Automation*, 2005, pp. 2381–2386.

[16] S. Wang and J. Braaksma, “Reinforcement learning control for biped robot walking on uneven surfaces,” in *the 2006 International Joint Conference on Neural Networks*, 2006, pp. 4173–4178.

[17] R. Tedrake, “Stochastic policy gradient reinforcement learning on a simple 3d biped,” in *Proc. of the 10th Int. Conf. on Intelligent Robots and Systems*, 2004, pp. 2849–2854.

[18] N. Kohl and P. Stone, “Policy gradient reinforcement learning for fast quadrupedal locomotion,” in *the IEEE International Conference on Robotics and Automation*, 2004, pp. 2619–2624.