2011 11th IEEE-RAS International Conference on Humanoid Robots
Bled, Slovenia, October 26-28, 2011

Online Learning of a Full Body Push Recovery Controller for Omnidirectional Walking

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

Abstract— **Bipedal humanoid robots are inherently unstable to external perturbations, especially when they are walking on uneven terrain in the presence of unforeseen collisions. In this paper, we present a push recovery controller for position-controlled humanoid robots which is tightly integrated with an omnidirectional walk controller. The high level push recovery controller learns to integrate three biomechanically motivated push recovery strategies with a zero moment point based omnidirectional walk controller. Reinforcement learning is used to map the robot walking state, consisting of foot configuration and onboard sensory information, to the best combination of the three biomechanical responses needed to reject external perturbations. Experimental results show how this online method can stabilize an inexpensive, commercially-available DARwin-OP small humanoid robot.**

Keywords: Bipedal Omnidirectional Walking, Full Body Push Recovery, Reinforcement Learning, Online Learning

I. Introduction

Due to their small footprint and high center of mass, bipedal humanoid robots are not statically stable. Open loop dynamic walking methods typically assume flat, well-defined surfaces for walking that ensure the pre-defined joint trajectories exhibit dynamic stability. However, these techniques can fail due to unevenness of the surface, collisions with obstacles, and unknown modeling errors of the robot or its actuators. Active stabilization is therefore crucial for operation of humanoid robots in realistic environments, and has been a topic of great interest in humanoid research.

There have been two main approaches proposed to handle active stabilization of humanoid robots. The first approach uses sensitive force sensors and an accurate full body model of the robot to calculate the proper joint torques required to reject any external disturbance forces [1], [2], [3], [4], [5]. The other is a biomechanically motived approach which focuses on fast, reactive push recovery behaviors using a simplified model of the robot [6], [7], [8], [9], [10], [11], [12].

One common drawback of these approaches is that most physical implementations require specialized hardware such as triaxial force and torque sensors, torque-controlled actuators, in addition to rapid computational processing and feedback control, making their implementation on inexpensive, commercially-available, position-controlled humanoid robots such as the Nao[1] or DARwIn-OP[2] infeasible.

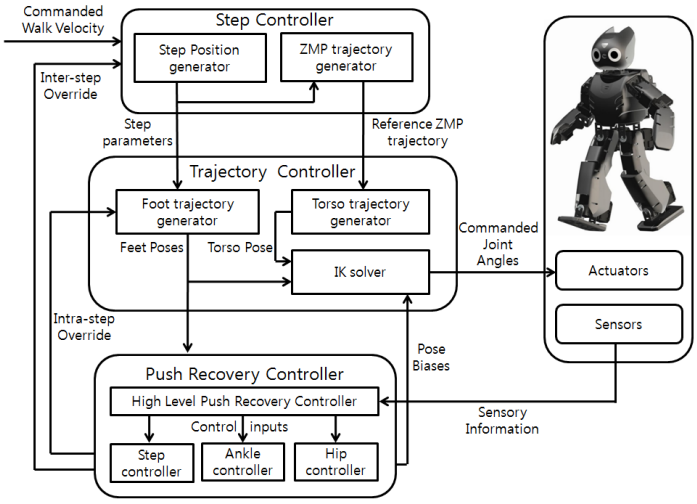


Fig. 1. Overview of the omnidirectional walk controller integrated with full body push recovery controller.

In previous work, we described a machine learning approach to learn a hierarchical full body push recovery controller for such hardware constrained humanoid robots [13]. In this paper, we address several shortcomings of the prior work. The first issue is that our previous work, along with most other humanoid push recovery implementations, assumes that the robot is either standing still or walking in place, and does not address more general situations when the robot is walking with nonzero velocity. The second issue is that a simulator was used to learn the controller, like other approaches utilizing machine learning [4], [5], which may not correctly model the robot and environmental properties.

Thus, the aim of this work is twofold. The first is to design an integrated walk controller with push recovery for position-controlled humanoid robots without force sensors that can stabilize the robot against external disturbances during omnidirectional walking. The second is to use the physical robot and experimental apparatus instead of a simulated environment to learn the push recovery controller parameters in an online fashion.

We first design a step-based omnidirectional walk controller that integrates with the biomechanically motivated push recovery controllers. Our method generates foot and

* 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

[1] http://www.aldebaran-robotics.com/
[2] http://www.robotis.com/xe/darwin_en

torso trajectories in real time based upon dynamic stability criteria, which can be switched to reject external perturbations. The current foot configuration and walking phase are taken into consideration to determine the control inputs for push recovery controllers. The robot is then used in conjunction with a motorized moving platform to apply a series of controlled perturbations to the robot. The parameters for the hierarchical controller are then learned in an online fashion using reinforcement learning. Experimental results show that our method can be successfully applied to the DARwin-OP robot platform.

The remainder of the paper proceeds as follows. Section II reviews three biomechanically motivated push recovery controllers and its implementation on position controlled humanoid robots. Section III explains the details of the step-based omnidirectional walk controller which is fully integrated with the push recovery controller. Section IV explains how the DARwIn-OP robot is used to learn the appropriate controller from experience in an online fashion. Section V shows the experimental results using the learned controller. Finally, we conclude with a discussion of outstanding issues and potential future directions arising from this work.

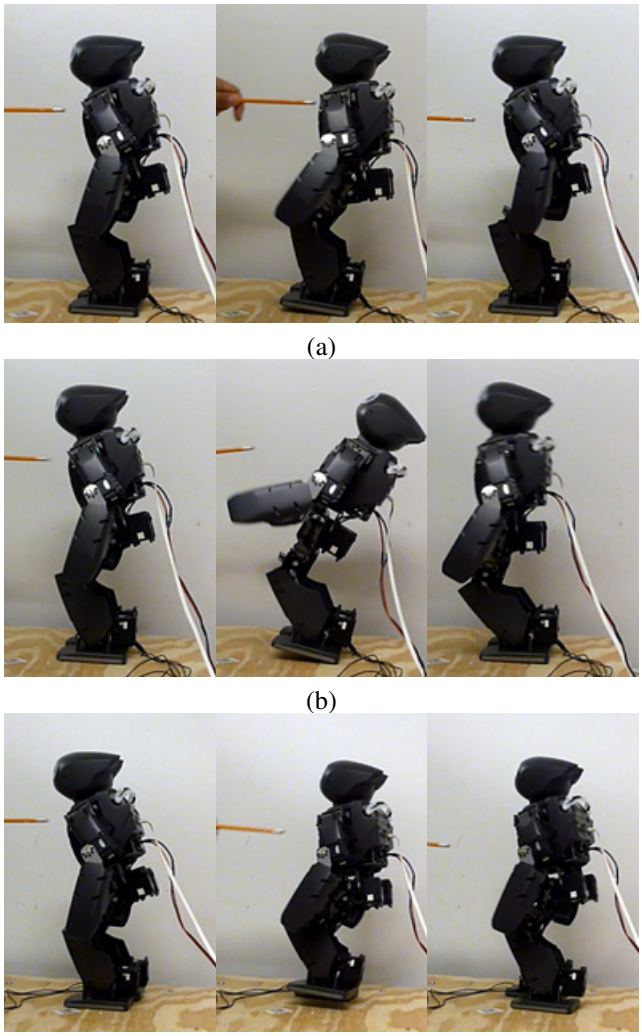
II. Biomechanically Motivated Push Recovery Control for Position Controlled Robots

Biomechanical studies show that humans display three distinctive motion patterns in response to sudden external perturbations: which we denote as ankle, hip and step push recovery strategies [7]. Although there has been some theoretical analysis of these strategies using simplified models, physical implementation of such analytical controllers on a generic, position controlled humanoid robot is not straightforward, and there has been little research on how to optimally mix these three strategies as humans do.

Instead of fully relying on analytic models, we suggested a machine learning approach to determine the appropriate push recovery controller which minimizes a predetermined cost function [14]. To generate the proper combination of push recovery strategies, a hierarchical high level controller uses current proprioceptive and inertial sensory information to determine the optimal combination of the three recovery strategies. In this section, we review the implementation of the three biomechanically motivated push recovery controllers on a position controlled robot, as shown in Figure 2.

A. Ankle push recovery

The ankle controller applies control torque on the ankle joints to keep the center of mass within the base of support. For position controlled actuators, controlling the ankle torque can be done by either controlling the auxiliary zero moment point (ZMP) which accelerates the torso to apply effective torque at ankle [13], [15] or modulating the target angle of the ankle servo. In this work we use the latter method, which is found to be effective for the particular robot we use in our experiments. Then the controller can be simply implemented



(a)

(b)

(c)

Fig. 2. Three biomechanically motivated push recovery strategies and corresponding controller for a position-controlled robot. (a) Ankle strategy that applies control torque on ankle joints. (b) Hip strategy which uses angular acceleration of torso and limbs to apply counteractive ground reaction force. (c) Step strategy which moves the base of support to a new position.

as

$$\Delta\theta_{ankle} = \quad x_{ankle} \tag{1}$$

where $\Delta\theta_{ankle}$ is the joint angle bias and x_{ankle} is the original ankle controller input. In addition to the ankle joints, we also modulate arm position to apply additional effective torque at the ankles in a similar way, unless overridden by the hip controller.

B. Hip push recovery

The hip controller uses angular acceleration of the torso and limbs to generate a backward ground reaction force to pull the center of mass back towards the base of support. For maximum effect, a bang-bang profile of the following form can be used

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

where $u(t)$ is the unit step function, τ_{max} is the maximum torque that the joint can apply, T_{R1} is the time the torso stops accelerating and T_{R2} is the time torso comes to a stop. After T_{R2}, the torso angle should return to the initial position. This two-stage control scheme can be approximately implemented with position controlled actuators as

$$\Delta\theta_{torso} = \begin{cases} x_{hip} & 0 \le t < T_{R2} \\ x_{hip}\frac{T_{R3}-t}{T_{R3}-T_{R2}} & T_{R2} \le t < T_{R3} \end{cases} \quad (3)$$

where $\Delta\theta_{torso}$ is the torso pose bias, T_{R3} the time torso angle completes returning to initial position, and x_{hip} is the input of hip controller. Arm rotation can also be used during hip push recovery to apply additional ground reaction force on the robot.

C. Step push recovery

The step controller effectively moves the base of support by taking a step. This is implemented by overriding the step controller to insert a new step with relative target foot position $x_{capture}$. In order to not lift the current support foot, the state of the robot needs to be monitored to determine the current foot configuration as well as the perturbation direction. Only during the appropriate walking phase is the walk control overridden with the new target foot position.

III. A Step-based Omnidirectional Walk Controller with Push Recovery Control

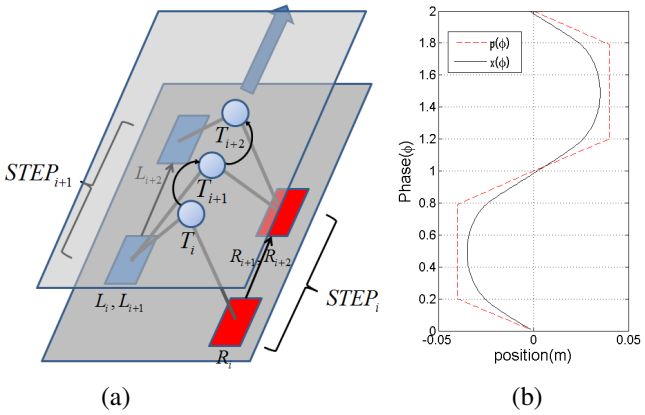
In this section, we describe the details of our omnidirectional walk controller which is integrated with the full body push recovery controller. As the active push recovery requires walk phase modulation and real-time modification of landing position, we cannot directly use a typical periodic time-based walk controller. Instead, we use a step-based walk controller that generates a series of steps with individual adjustable parameters such as duration, support foot selection and landing position. The overall walk control is separated into a step controller and trajectory controller, where the step controller determines the parameters for each step and the trajectory controller generates foot and torso position trajectories based upon that information. In this section we cover each controller in detail, and show how they are integrated with the high level push recovery controller described in the previous section.

A. Step controller

The step controller determines the parameters of each step, which is defined as

$$STEP_i = \{SF, WT, t_{STEP}, L_i, T_i, R_i, L_{1+1}, T_{i+1}, R_{i+1}\} \quad (4)$$

where SF denotes the support foot, WT denotes the type of the step, t_{STEP} is the duration of the step, and L_i, T_i, R_i, and $L_{i+1}, T_{i+1}, R_{i+1}$ are the initial and final 2D poses of left foot, torso and right foot in (x, y, θ) coordinate. L_i, T_i, R_i are determined by the final feet and torso poses from the last step, and L_{i+11}, R_{i+1} is calculated using the command walk velocity and current foot configuration to enable omnidirectional walking. Foot reachability and self-collision constraints are also taken into account when calculating target foot poses. Target torso pose T_{i+1} is set to the middle point of L_{i+1} and R_{i+1} by default, so that the transition occurs at the most stable posture where the center of mass (CoM) lies on the midpoint of the two support positions.



(a) (b)

Fig. 3. The step-based walk controller. (a) An example of walking behavior which is composed of two steps, $STEP_i$ and $STEP_{i+1}$. (b) Corresponging ZMP and torso trajectory p and x. Timing parameters of $\phi_1 = 0.2$ and $\phi_2 = 0.8$ are used.

Once we have the initial and final poses of the feet and torso, we can define the reference ZMP trajectory $p_i(\phi)$ as the following piecewise-linear function for the left support case

$$p_i(\phi) = \begin{cases} T_i(1-\frac{\phi}{\phi_1}) + L_i\frac{\phi}{\phi_1} & 0 \le \phi < \phi_1 \\ L_i & \phi_1 \le \phi < \phi_2 \\ T_{i+1}(1-\frac{1-\phi}{1-\phi_2}) + L_i\frac{1-\phi}{1-\phi_2} & \phi_2 \le \phi < 1 \end{cases} \quad (5)$$

or for the right support case

$$p_i(\phi) = \begin{cases} T_i(1-\frac{\phi}{\phi_1}) + R_i\frac{\phi}{\phi_1} & 0 \le \phi < \phi_1 \\ R_i & \phi_1 \le \phi < \phi_2 \\ T_{i+1}(1-\frac{1-\phi}{1-\phi_2}) + R_i\frac{1-\phi}{1-\phi_2} & \phi_2 \le \phi < 1 \end{cases} \quad (6)$$

where ϕ is the walk phase and ϕ_1, ϕ_2 are the timing parameters determining the transition between single support and double support phase.

Finally, the step controller allows for inter-step override from the push recovery controller. The hip and step push recovery controller can temporarily stop the walking to stabilize the robot after each push recovery behavior is initiated, and the step push recovery controller can initiate a special capture step to reject large perturbations. The walk type parameter WT is used to describe these special types of steps.

B. Trajectory controller

The trajectory controller generates foot and torso trajectories for the current step defined in (4). First we define the single support walk phase ϕ_{single} as

$$\phi_{single} = \begin{cases} 0 & 0 \le \phi < \phi_1 \\ \frac{\phi-\phi_1}{\phi_2-\phi_1} & \phi_1 \le \phi < \phi_2 \\ 1 & \phi_2 \le \phi < 1 \end{cases} \quad (7)$$

then we use following parameterized trajectory function

$$f_x(\phi) = \phi^{\alpha} + \beta\phi(1-\phi) \tag{8}$$

to generate the foot trajectories

$$L_i(\phi_{single}) = L_{i+1} f_x(\phi_{single}) + L_i(1 - f_x(\phi_{single})) \tag{9}$$

$$R_i(\phi_{single}) = R_{i+1} f_x(\phi_{single}) + R_i(1 - f_x(\phi_{single})) \tag{10}$$

The torso trajectory $x_i(t)$ is calculated according to the following ZMP criterion for a linear inverted pendulum model

$$p = x - t_{ZMP}\ddot{x} \tag{11}$$

The piecewise linear ZMP trajectory we use in (5), (6) yields for $x_i(\phi)$ with zero ZMP error during the step period

$$x_i(\phi) = \begin{cases} p_i(\phi) + a_i^p e^{\phi/\phi_{ZMP}} + a_i^n e^{-\phi/\phi_{ZMP}} \\ \quad -\phi_{ZMP} m_i \sinh \frac{\phi-\phi_1}{\phi_{ZMP}} & 0 \le \phi < \phi_1 \\ p_i(\phi) + a_i^p e^{\phi/\phi_{ZMP}} + a_i^n e^{-\phi/\phi_{ZMP}} & \phi_1 \le \phi < \phi_2 \\ p_i(\phi) + a_i^p e^{\phi/\phi_{ZMP}} + a_i^n e^{-\phi/\phi_{ZMP}} \\ \quad -\phi_{ZMP} n_i \sinh \frac{\phi-\phi_2}{\phi_{ZMP}} & \phi_2 \le \phi < 1 \end{cases} \tag{12}$$

where $\phi_{ZMP} = t_{ZMP}/t_{STEP}$ and m_i, n_i are ZMP slopes which are defined as follows for the left support case

$$m_i = (L_i - T_i)/\phi_1 \tag{13}$$

$$n_i = -(L_i - T_{i+1})/(1-\phi_2) \tag{14}$$

and for the right support case

$$m_i = (R_i - T_i)/\phi_1 \tag{15}$$

$$n_i = -(R_i - T_{i+1})/(1-\phi_2) \tag{16}$$

Then the parameters a_i^p and a_i^n can be uniquely determined by the boundary condition $x_i(0) = T_i$ and $x_i(1) = T_{i+1}$. This analytic solution has zero ZMP error during the step period but discontinuous jerk at the transitions. However, as this transition occurs in the middle of the most stable double support stance, we found this does not hamper stability.

In addition to calculating foot trajectories based upon predetermined target foot poses, the intra-step override from the push recovery controller can also modify the foot landing position while stepping. When the step push recovery is triggered during the initial phase of walking, the target landing position is overridden and a new foot trajectory is generated under foot reachability constraints and foot maximum velocity constraints.

C. Push recovery controller

The push recovery controller consists of three low level biomechanically motivated push recovery strategies and a high level controller that generates the control inputs for them based on sensory feedback and current state information. There is no closed-form analytic solution for combining the individual low level controllers, so we use a machine-learning approach that trains the overall controller to optimize a cost function from experience. We formalize the high level controller as a reinforcement learning problem with the following state

$$S = \left\{ \theta_{IMU}, \theta_{gyro}, \theta_{foot}, \phi, FD, HCS \right\} \tag{17}$$

where θ_{IMU} and θ_{gyro} are the torso pose and gyroscope data from inertial sensors, θ_{foot} is the support foot pose calculated using forward kinematics and onboard sensors, ϕ is the walk phase, FD is the foot displacement, and HCS is the hip controller state. The learned action is defined as the combined inputs for the three low level controllers

$$A = \left\{ x_{ankle}, x_{hip}, x_{capture} \right\} \tag{18}$$

and the reward is defined as

$$R = \left| \theta_{gyro} \right|^2 + \frac{g}{z_0} \left| \theta_{IMU} \right|^2 \tag{19}$$

where g is the gravitational constant and z_0 is the COM height.

IV. Online learning of the push recovery controller

We have implemented the integrated walk controller with push recovery on a commercially available small humanoid robot and trained the controller in an online fashion. In this section, we present the details of the learning setup.

A. Hardware setup

We use the commercially available DARwIn-OP humanoid robot developed by Robotis Co., Ltd. and the RoMeLa laboratory. It is 45 cm tall, weighs 2.8kg, and has 20 degrees of freedom. It has a USB camera for visual feedback, and 3-axis accelerometer and 3-axis gyroscope for inertial sensing. Position-controlled Dynamixel servos are used for actuators, which are controlled by a custom microcontroller connected by an Intel Atom-based embedded PC at a control frequency of 100Hz.

To generate repeatable external perturbations, a motorized moving platform was constructed using Dynamixel servo-motors. The motorized platform is attached to the same bus connecting the actuators of the robot, and controlled by the same I/O process. To generate maximum peak acceleration, the platform is slowly accelerated in one direction and then suddenly accelerated in the opposite direction. We have found the platform can generate accelerations greater than 0.5g while carrying the robot, providing large enough perturbations to make the robot fall.

B. Learning setup

In previous work [13], a policy gradient reinforcement learning algorithm was used to learn the push recovery controller in a simulated environment. However, the same approach cannot be directly applied to the physical robot for online learning. The main issue with machine learning approaches on physical robots is that the amount of training data is severely restricted. Unlike in simulation, small humanoid robots cannot continuously operate for long periods of time due to heat buildup at the actuators, and the process of gathering training data is slower and may require

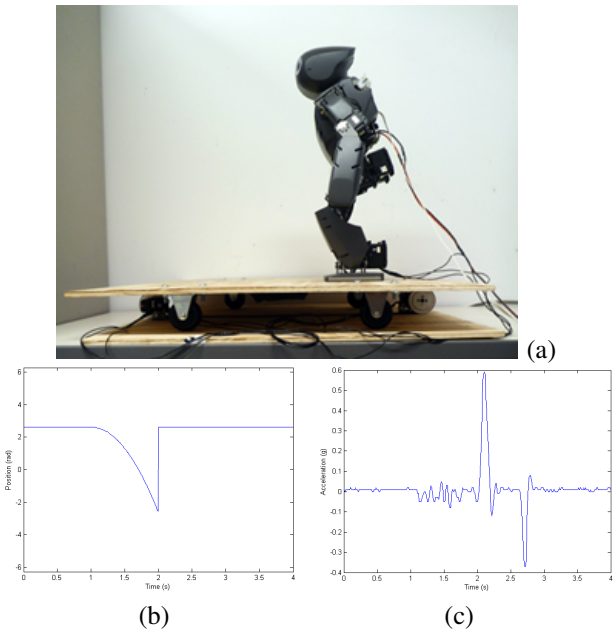


Fig. 4. (a) Learning setup including the servo controlled platform connected to the DARwIn-OP robot. (b) Commanded position of the platform to induce instantaneous disturbance. (c) Actual acceleration of the platform.

more human intervention. Also, the sensory noise present in physical experiments are not well-modeled in simulations.

To overcome these problems, we simplify the reinforcement learning formulation of the high level controller as follows. First we use the heuristic disturbance variable θ_d as a single continuous state variable which is defined as

$$\theta_d = s(\theta_{IMU} + k_{gyro}\theta_{gyro}) \tag{20}$$

where s is a moving average smoothing function and k_{gyro} is a heuristic mixing parameter. The rest of the state variables are discretized to form the discrete state variables

$$S_d = \left\{\hat{\phi}, \hat{FD}, \hat{HCS}\right\}. \tag{21}$$

Simple parametric functions with preset parameters k_{ankle}, k_{hip}, k_{step} are then used to describe the possible controller actions:

$$x_{ankle} = \begin{cases} 0 & |\theta_d| < \theta^{TH}_{ankle}(S_d) \\ k_{ankle}(\theta_d - \theta^{DB}_{ankle}(S_d)) & \theta_d \leq -\theta^{TH}_{ankle}(S_d) \\ k_{ankle}(\theta_d + \theta^{DB}_{ankle}(S_d)) & \theta_d \geq \theta^{TH}_{ankle}(S_d) \end{cases} \tag{22}$$

$$x_{hip} = \begin{cases} 0 & |\theta_d| < \theta^{TH}_{hip}(S_d) \\ -k_{hip} & \theta_d \leq -\theta^{TH}_{hip}(S_d) \\ k_{hip} & \theta_d \geq \theta^{TH}_{hip}(S_d) \end{cases} \tag{23}$$

$$x_{capture} = \begin{cases} 0 & |\theta_d| < \theta^{TH}_{step}(S_d) \\ -k_{step} & \theta_d \leq -\theta^{TH}_{step}(S_d) \\ k_{step} & \theta_d \geq \theta^{TH}_{step}(S_d) \end{cases} \tag{24}$$

Finally, to reduce the effects of incorrect actions during normal walking, a negative training set is generated by letting the robot walk without any external perturbations on the platform to prevent the controller from learning overly sensitive corrections.

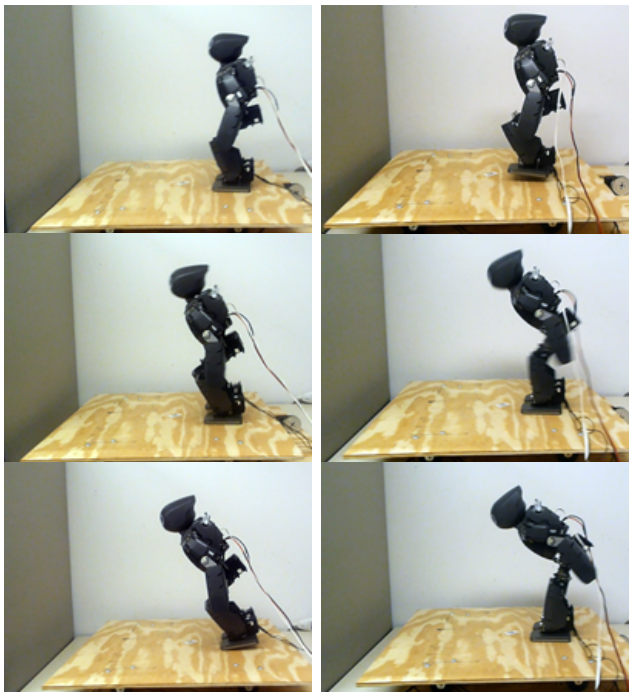


Fig. 5. Learning process of the push recovery controller. (a) Initial parameter set which does not trigger hip strategy. (b) Learned parameter set after training that triggers hip strategy when large amount of disturbance is applied. The same amount of disturbance is applied to the robot.

V. Experimental Results

A. Learning the push recovery controller

We trained the push recovery controller using the servo controlled platform. The robot is set to walk in different directions, and a platform movement is triggered at different phases of the walking. Then the robot is repositioned by the human operator to start a new trial. Each trial takes roughly 10 seconds on average, and 20 trials are done per episode to improve the controller. Figure 5 shows a comparison of the robot behavior before and after 20 episodes of training with the same amount of perturbation. We can see that the robot learns to apply the appropriate push recovery behavior to reject large perturbations during walking.

B. Testing the push recovery controller

We let the robot with learned controller freely walk over a flat surface and applied a number of pushes with various directions and magnitude. Figure 6 shows some examples of robot response against external disturbances[3]. We see that the learned controller can successfully trigger appropriate push recovery behaviors to keep the omnidirectionally walking robot from falling down.

VI. Conclusions

We have demonstrated an integrated controller that enables full body push recovery during omnidirectional walking

[3] http://www.youtube.com/watch?v=N5Sh407Mqjg

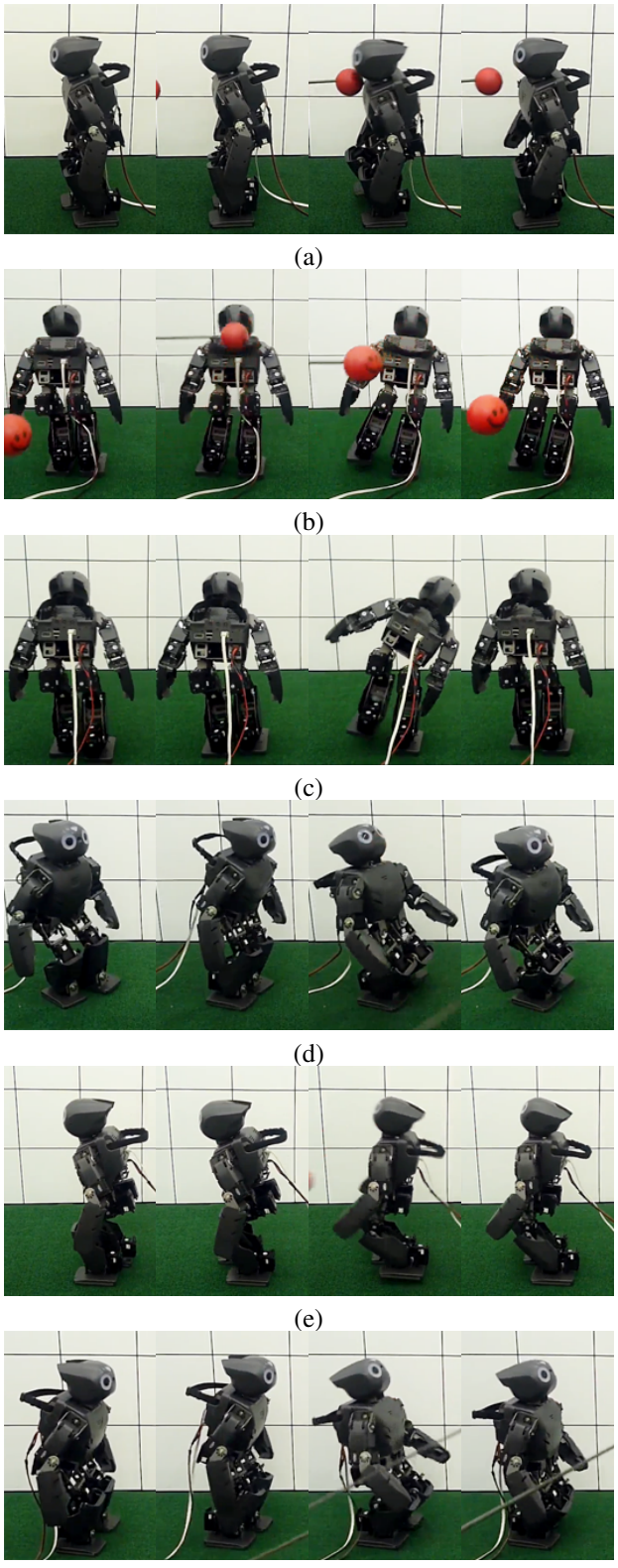


Fig. 6. Responses of the learned push recovery controller against perturbation during omnidirectional walking. (a) step strategy during walking forward. (b) step strategy during sidestepping. (c) hip strategy during walking forward. (d) hip strategy during turning. (e) hip and step strategies during walking forward. (f) hip and step strategies during sidestepping.

for humanoid robots without specialized sensors and actuators. Three low level biomechanically motivated push recovery strategies are implemented on a position controlled humanoid robot, and integrated with a non-periodic, step-based omnidirectional walk controller. The proper parameters for the hierarchical controller are learned online using a commercially-available small humanoid robot along with a servo-controlled moving platform. Experimental results show that the trained controller can successfully initiate a full body push recovery behavior under external perturbations while walking in an arbitrary direction.

Possible future work includes incorporating more sophisticated learning algorithms to utilize the limited training data, and implementing these algorithms on full-size humanoid robots.

ACKNOWLEDGMENTS

We acknowledge the support of the NSF PIRE program under contract OISE-0730206. This work was also partially supported by the IT R&D program of MKE/KEIT (KI002138, MARS), the industrial strategic technology development program (10035348) funded by the Korean government (MKE), and the BK21-IT program.

REFERENCES

[1] S.-H. Hyon and G. Cheng, "Disturbance rejection for biped humanoids," *ICRA*, 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] S.-H. Hyon, R. Osu, and Y. Otaka, "Integration of multi-level postural balancing on humanoid robots," in *ICRA*, 2009, pp. 1549–1556.
[4] T. Matsubara, J. Morimoto, J. Nakanishi, S.-H. Hyon, J. G. Hale, and G. Cheng, "Learning to acquire whole-body humanoid center of mass movements to achieve dynamic tasks," *Advanced Robotics*, vol. 22, no. 10, pp. 1125–1142, 2008.
[5] A. Yamaguchi, S.-H. Hyon, and T. Ogasawara, "Reinforcement learning for balancer embedded humanoid locomotion," in *Humanoids*, dec. 2010, pp. 308 –313.
[6] B. Jalgha, D. C. Asmar, and I. Elhajj, "A hybrid ankle/hip preemptive falling scheme for humanoid robots," in *ICRA*, 2011.
[7] M. Abdallah and A. Goswami, "A biomechanically motivated two-phase strategy for biped upright balance control," in *ICRA*, 2005, pp. 2008–2013.
[8] J. Pratt, J. Carff, and S. Drakunov, "Capture point: A step toward humanoid push recovery," in *Humanoids*, 2006, pp. 200–207.
[9] T. Komura, H. Leung, S. Kudoh, and J. Kuffner, "A feedback controller for biped humanoids that can counteract large perturbations during gait," in *ICRA*, 2005, pp. 1989–1995.
[10] 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.
[11] B. Stephens, "Humanoid push recovery," in *Humanoids*, 2007.
[12] B.-K. Cho and J.-H. Oh, "Practical experiment of balancing for a hopping humanoid biped against various disturbances," in *IROS*, 2010, pp. 4464–4470.
[13] S.-J. Yi, B.-T. Zhang, and D. D. Lee, "Online learning of uneven terrain for humanoid bipedal walking." in *AAAI'10*, 2010.
[14] S.-J. Yi, B.-T. Zhang, D. Hong, and D. D. Lee, "Learning full body push recovery control for small humanoid robots." in *ICRA*, 2011.
[15] S. Kajita, M. Morisawa, K. Harada, K. Kaneko, F. Kanehiro, K. Fujiwara, and H. Hirukawa, "Biped walking pattern generator allowing auxiliary zmp control," in *IROS*, oct. 2006, pp. 2993 –2999.