Learning Capture Points for Humanoid Push Recovery

John Rebula, Fabián Cañas, Jerry Pratt
The Institute for Human and Machine Cognition
Pensacola, FL 32502
Email: jrebula@alum.mit.edu

Ambarish Goswami
Honda Research Institute
Mountain View, CA 94041
Email: agoswami@honda-ri.com

***Abstract*— We present a method for learning Capture Points for humanoid push recovery. A Capture Point is a point on the ground to which the biped can step and stop without requiring another step. Being able to predict the location of such points is very useful for recovery from significant disturbances, such as after being pushed. While dynamic models can be used to compute Capture Points, model assumptions and modeling errors can lead to stepping in the wrong place, which can result in large velocity errors after stepping. We present a method for computing Capture Points by learning offsets to the Capture Points predicted by the Linear Inverted Pendulum Model, which assumes a point mass biped with constant Center of Mass height. We validate our method on a three dimensional humanoid robot simulation with 12 actuated lower body degrees of freedom, distributed mass, and articulated limbs. Using our learning approach, robustness to pushes is significantly improved as compared to using the Linear Inverted Pendulum Model without learning.**

I. Introduction

Appropriate foot placement is critical when attempting to come to a stop during dynamic three dimensional bipedal walking. Simple models can be used to determine where to step in order to stop [9], [26], but the errors resulting from the assumptions of these models may be significant, especially in bipeds with a large number of degrees of freedom, distributed mass, and complex walking control systems. More complex models can reduce the errors resulting from these assumptions, but even with accurate models, small errors in modeling parameters may lead to significant errors in predicting the desired stepping location.

Humans are very adept at stepping in just the right place after being pushed in order to regain balance. From simple observations of toddlers it is clear that this ability improves with practice and that learning likely plays a large role in a person's ability to choose an appropriate place to step. This ability becomes almost reflexive as can be experienced by the following simple demonstration:

Stand on one leg and have someone push you in a random direction. Notice how you step to a spot that allows you to stand on one leg again. Now, still standing on one leg, have someone push you again in a random direction, but this time purposefully step to a spot on the ground that does not allow for you to regain balance and stop. Notice how natural and automatic it seems when you step in a good spot but how difficult and how much conscious effort it takes to step in a bad spot.

Since humans are so adept at stepping to an appropriate place to recover from pushes, we argue that humanoid robots need to become just as adept at this skill in order to operate in real world environments and perform tasks that humans perform.

Learning where to step in order to recover from a disturbance is a promising approach that does not require sophisticated modeling, accurate physical parameter measurements, or overly constraining limitations on the walking control system. In this paper we present a method for learning Capture Points for humanoid push recovery. Loosely speaking, a Capture Point is a point on the ground in which a biped can step to and stop without requiring another step [26], [28]. The locus of all possible Capture Points is the Capture Region.

The location of the Capture Region depends on a number of factors, including the state of the robot and the kinematic reachability of the swing leg. In addition, one can define Capture Points with certain qualifications or assumptions. In this paper we assume the robot has an existing control system which can step to any reasonable desired location on the ground, and it takes this desired location as an input variable. Given the robot and control system, we learn the desired stepping location in order for the robot to stop. This method should be applicable to any humanoid robot and control system that has certain characteristics described in the next section.

In this paper we validate our method on a three dimensional Humanoid robot simulation with 12 actuated lower body degrees of freedom. Figure 1 shows simulation results before learning and after learning for a given push. Before learning the robot fails to recover in a single step, while after learning it does recover.

II. Problem Statement

We wish to learn Capture Points, points on the ground that the robot can step to and stop in a single step. These Capture Points are a function of the robot's state, the control actions during the current stance phase, and the control actions during the next stance phase in which the robot attempts to stop. In this paper we assume that the robot's control systems already exist and that we can modify the module, which we call the Desired Stepping Location Calculator, that determines the desired stepping location. We then use learning to improve this

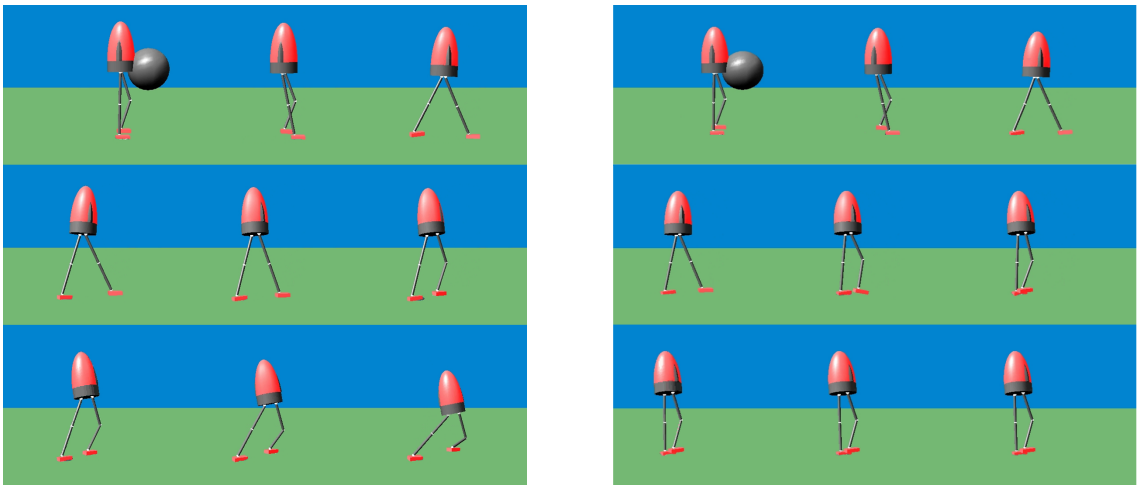

Fig. 1. Simulation results before learning (left) and after learning (right) showing the humanoid robot reacting to a disturbance in a single step. Before learning the robot fails to recover balance. After learning the robot successfully comes to a stop in one step. The ball shown in the first frame of each animation applied a push from straight on at 550 Newtons for 0.05 seconds. Images are from left to right, top to bottom and are spaced 0.04 seconds apart.

module, better predicting Capture Points, given the preexisting control systems.

Before stating the problem, we first define a few terms. Our goal is for the robot to come to a stop after being pushed. We define a Stopped State as follows:

- **Stopped State**: Robot state in which the robot's Center of Mass velocity, angular momentum about the Center of Mass, and Center of Mass linear and rotational accelerations are all less than given thresholds.

Note that the velocity and acceleration thresholds need to be non-zero for Stopped States to exist for many bipedal models. For example, for a point-foot point-mass biped, zero Center of Mass velocity and acceleration cannot be achieved during single support in the presence of infinitesimal disturbances. With small but finite thresholds, the robot will be essentially stopped, such that with a small amount of Center of Pressure regulation on the foot or a small amount of upper body motions, the robot could maintain balance without requiring any additional steps. For humanoids with non-point feet, given reasonable velocity and acceleration thresholds, being in a Stopped State will also imply that the Capture Region has a large intersection with the convex hull of the support feet.

We consider the gait of the robot to have two main phases, the Capture Step and the Stopping Stance Phase:

- **Capture Step**: The single support phase and swing foot placement during which the robot takes a step in attempts to reach a Stopped State during the next support phase.
- **Stopping Stance Phase**: The support phase occurring after the Capture Step, during which the robot attempts to reach a Stopped State, without taking any additional steps.

Note that the Stopping Stance Phase could be a double support phase, followed by a single support phase, or could be an instantaneous exchange of support into a single support phase. We define the control systems that are in effect during each of these two stages of the gait as follows:

- **Stepping Control System**: The control system the robot utilizes during the Capture Step. π_{step}.
- **Stopping Control System**: The control system the robot utilizes during the Stopping Stance Phase. π_{stop}.

The Stepping Control System will determine the location that the robot steps to. We define the module of this control system that determines the desired location to step to as the Desired Step Location Calculator,

- **Desired Step Location Calculator**: Control system module of the Stepping Control System that calculates the desired step location in attempts to reach a Stopped State during the Stopping Stance Phase.

Note that the Stepping Control System may query the Desired Step Location Calculator either at predefined trigger events, or continuously throughout the swing, to determine where to step to. Let us make the following assumptions:

A1 The Stepping Control System is a function of the robot's state, $\vec{x}$, some internal controller state variables, $\vec{p}_{step}$, and a desired stepping location, $\vec{x}_{step}$. $\vec{u} = \pi_{step}(\vec{x}, \vec{p}_{step}, \vec{x}_{step})$. The Stepping Control System does a reasonable job of stepping to the desired step location, maintains balance, and regulates any other important aspects of walking.

A2 The Stopping Control System is a function of the robot's state, $\vec{x}$, and perhaps some internal controller state variables, $\vec{p}_{stop}$. $\vec{u} = \pi_{stop}(\vec{x}, \vec{p}_{stop})$. The Stopping Control System does a reasonable job of stopping (reaching a Stopped State), without requiring an additional step, from a reasonably sized basin of attraction of states.

Our problem statement is then as follows: Given a biped with a preexisting Stepping Control System and a preexisting Stopping Control System, determine the desired foot placement for the Stepping Control System, such that after the robot takes a step, the Stopping Control System will bring the robot to a Stopped State. In other words, determine a Desired

Step Location Calculator $\vec{x}_{step} = \pi_{capture}(\vec{x}, \vec{p}_{step}, \vec{p}_{capture})$, where $\vec{p}_{capture}$ represents potential internal state of the Desired Step Location Calculator. Note that we assume the Desired Step Location Calculator has access to both the state of the robot, $\vec{x}$, and any internal state of the Stepping Controller, $\vec{p}_{step}$.

In this paper we assume that a Stepping Control System and a Stopping Control System already exist for the biped so that we can concentrate on using learning to determine a Desired Step Location Calculator. We emphasize the Step Location Calculator because choosing an appropriate stepping location is perhaps the most critical control action to take when attempting to stop in bipedal walking, particularly in 3D bipedal walking. Other control actions, such as moving the Center of Pressure on the feet, accelerating internal momentum (e.g., lunging the upper body or "windmilling" the arms), or altering the Center of Mass trajectory are also useful control actions. However, their success in push recovery depends on the robot first choosing an appropriate step location, such that these control actions can be used to bring the robot to a Stopped State.

In the experiments for this paper, the Stepping Control System and the Stopping Control System that we employ each are designed using simple physics based heuristic control rules [25]. The main strategy employed for the stopping control system is to modulate the Center of Pressure on the support foot based on the Linear Inverted Pendulum Model in order to have the Center of Mass come to rest over the foot. In this paper, we do not use strategies that involve accelerating internal momentum or altering the Center of Mass trajectory based on velocity. However, controllers that use these strategies could also be used in the presented learning method and should result in larger Capture Regions than controllers that do not employ these strategies.

We state the above problem fairly generally and would like to learn Capture Regions for any state of the robot. As a stepping stone to that result, in this paper we focus on the problem of starting in a Stopped State, being pushed, and then taking a step and returning to a Stopped State. The robot starts balancing over one leg with the other leg raised off the ground. A pushing force occurs, disrupting the robot's balance and requiring the robot to take a step to stop. Setting the robot up like this initially is less general, but it allows us to focus on this particular instance of the problem, and allows us to generate a sufficient set of data to learn from with a reasonable number of trials. In addition, once this skill is learned, it can be used as the basis for stopping from more general conditions.

III. Simulation Model

The learning method is verified on a simulated 12 degree of freedom humanoid robot. Table I details the important parameters of the model.

IV. Learning Techniques

We present two different learning techniques, one off-line and one on-line, to solve the problem presented in the previous section. Instead of learning from scratch, both techniques employ a nominal stepping location as a starting point to learn from. The nominal stepping location is based on the Linear Inverted Pendulum Model [9] of walking and, in our simulations, typically predicts a Capture Point well within 20 centimeters of an actual Capture Point. The off-line learning technique represents the learned Capture Point as an offset from the Linear Inverted Pendulum Capture Point. This offset is calculated based on curves fit to several variables of the state vector. The on-line learning technique represents the learned Capture Point using a Radial Basis Function Neural Network.

A. Stopping Energy Error

To evaluate the performance of the learned Desired Step Location Calculator, it will be necessary to compare the errors resulting from different stepping locations. While there is a clear distinction between a Stopped State and a state that will lead to a fall, a measure of stopping error is required to compare two falling states.

To compare results from stepping in different locations, we define a Residual Stopping "Energy" as,

$$E_{stopping} = \frac{1}{2}|\dot{\vec{x}}|^2 + \frac{1}{2}\frac{g}{l}|\vec{x}|^2, \quad (1)$$

where $\vec{x}$ is the ground projection of the vector from the support foot to the Center of Mass, g is the gravitational acceleration, and l is the average Center of Mass height. Note that this quantity is similar to the conserved Orbital Energy of the Linear Inverted Pendulum Model [9], except that instead of subtracting the two terms, we add them.

$E_{stopping}$ will decrease as the Center of Mass approaches the foot and decelerates. It will reach a minimum approximately when the Center of Mass is the closest to the foot. We use the minimum value achieved during an attempted stop as the Stopping Error, and refer to the state of the robot at that instance as the "Top of Stance" state. For a given push, it is reasonable to expect that the Stopping Error will decrease as the robot steps closer and closer to the Capture Region. When the robot steps to a Capture Point, the Stopping Error will be near zero since the robot will pass through a Stopped State, with the Center of Mass approaching a balanced position over the foot.

B. Off-line Learning Technique

In the off-line learning technique, we collect data resulting from a number of different pushes on the robot, and a number

TABLE I
Simulated Robot Parameters

Total Mass	28.84 kg
Mass in Body	16.0 kg
Mass in Each Leg	6.42 kg
Leg Length	0.911 m
Standing Center of Mass Height	0.945 m
Foot Width	0.1 m
Foot Length	0.203 m

of different stepping locations on the ground. The pushes were simulated as impact forces on the robot for a set time, 0.05 sec, at varying magnitudes, 150 to 550 N at 100 N intervals, yielding force impulses between 7.5 and 27.5 $Nsec$. The direction of the applied force was varied over the full range of [0, 2π] at 10 even intervals. During learning, the Desired Step Location Calculator returned the Capture Point predicted by the Linear Inverted Pendulum Model plus an offset that varies along a 15 by 15 point grid spanning $\pm 0.15m$ in both x and y. For each push setting (magnitude and direction pair), and offset, the robot is started from rest on one foot and disturbed by the impact force. The Desired Step Location Calculator decides where to step, and the simulation continues until the next support phase either reaches a Stopped State or the Center of Mass falls below $0.65m$, indicating a fall. Each force setting is simulated for 225 swings, one for each offset. There are 500 force settings, yielding 112500 trials.

The centroid of the Capture Region for each force setting is then calculated from the trial data. This yields an offset vector from the nominal location estimated by the Linear Inverted Pendulum Model to the centroid of the Capture Region. We learn a Desired Step Location Calculator by fitting functions of the decision state to estimate the offset vector to the centroid of the Capture Region. It was found that the angle and magnitude of the desired offset were best correlated with the angle (in the XY plane) of the Center of Mass velocity at the decision event, so this was the only dimension of the state vector used in the calculation of the offset. The learned Desired Step Location Calculator determines the desired step by calculating the Inverted Pendulum Capture Point and adding an offset vector calculated by evaluating the fit functions for the offset magnitude and angle using the decision state Center of Mass velocity angle.

To test the Desired Step Location Calculator, a new set of pushes was generated at a finer resolution, 150 to 550 N at 50 N intervals and 20 even angle intervals. These pushes were simulated on the robot using both the simple Linear Inverted Pendulum model to calculate Capture Points and the learned Desired Step Location Calculator.

C. On-line Learning Technique

In the on-line learning technique, the robot is pushed randomly and the Desired Step Location Calculator is used to decide where to step. Based on the results of the step, the Desired Step Location Calculator is updated. The simulation is run for 4000 random pushes. During the first 1000 trials, learning is turned off in order to get a baseline estimate of success rate. Learning occurs during trials 1000-3000. During the final 1000 trials learning is turned off again in order to get a final estimate of post-learning success rate. In each of the trials, the push force and direction is randomly determined. The force magnitude varies in the range of 200 to 450 Newtons, yielding force impulses between 10 and 22.5 $Nsec$. The force direction varies in the range $[-\frac{\pi}{4}, \frac{5\pi}{4}]$. The angle is measured from the forward axis of the biped. We omitted the $\frac{\pi}{2}$ radian sector to the right of the robot because the controller currently has difficulty maintaining stability in that region, and it forces the swing leg (the left leg) to cross through the support leg.

1) Radial Basis Function Network: The Desired Step Location Calculator stores learned Capture Points in a set of 11 Radial Basis Function Neural Networks. Given a state, the Calculator chooses the Network to use based on the distance in the XY plane of the Center of Mass from the support foot. Each network has a two dimensional input vector composed of the Linear Inverted Pendulum Capture Point x and y location with respect to the support foot.

Each Radial Basis Function Network has 4356 Gaussian basis functions with centers on a grid with 3cm spacing along each input dimension. Each Gaussian basis function is defined by

$$G(\vec{x}) = Ae^{(-\frac{(\vec{x}-\vec{x_o})^2}{r^2})} \tag{2}$$

where r is $0.03m$, $\vec{x}$ is the input velocity vector, and $\vec{x_o}$ is the center of the Gaussian. A is the weight attributed to the basis function, and is updated to train the network.

When the controller queries for a Capture Point, the Desired Step Location Calculator calculates the Linear Inverted Pendulum Capture Point, $(x_{c,lip}, y_{c,lip})$ from the current state of the robot. It then selects the Network to use based on the distance of the Center of Mass from the support foot in the Cartesian plane. Each Network is covers a range of $0.03m$ of Center of Mass distance. It then uses $(x_{c,lip}, y_{c,lip})$ as the input vector to the Radial Basis Function Network. Based on the result of the step, the network is then updated.

2) Learning Update Rule: After each swing trial, if the step location succeeded in bringing the robot to a Stopped State, no update is applied to the network. If the step location chosen resulted in the robot falling, the trial is stopped when the body height goes below a threshold, 0.85 m. This final velocity vector of the body as it falls away from the support foot corresponds to the appropriate direction in which to correct the Desired Step Location. Therefore the desired offset, $\vec{\delta}_{offset}$, is calculated as

$$\vec{\delta}_{offset} = E_{stopping}\dot{x}_{final} \tag{3}$$

where the magnitude to correct the location by is determined by Equation 1, the energy error of the top of stance. A learning gain is then applied to maintain stability of the learning process, yielding

$$\Delta V = \delta_{offset} K_{learning}, \tag{4}$$

where $K_{learning} = 0.6$.

The x and y values of ΔV are used to update the weights of each Radial Basis Function. This is done by calculating the total unweighted response of the set of basis functions $\mathbf{G}$ in each network, $V_{unweighted}$ by setting all the function weights to 1,

$$V_{unweighted} = \sum_{G \in \mathbf{G}} G(\vec{x})\big|_{A=1}. \tag{5}$$

Each weight in the network is changed by calculating its unweighted contribution to the network and multiplying by the desired ΔV.

$$\Delta A_i = \Delta V \frac{G_i|_{A=1}}{V_{unweighted}}. \quad (6)$$

V. Simulation Results

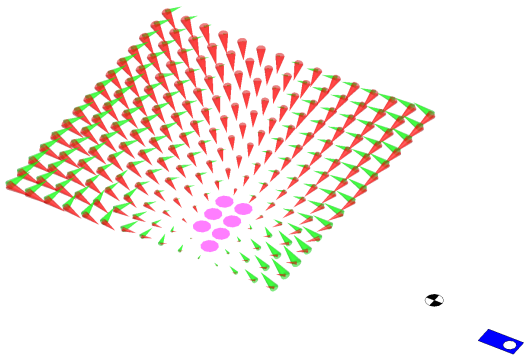

Fig. 2. Diagram showing results of stepping to various locations for a given push. Pink circles show Capture Points. At other points a red arrow shows the resulting vector from the new support foot to the Center of Mass Position and a green arrow shows the resulting Center of Mass Velocity at the point where the Center of Mass comes closest to the center of the support ankle. The black and white target shows the ground projection of the Center of Mass when the Desired Step Location Calculator was queried. The blue rectangle shows the location of the current stance foot. The center of the grid of points is the location of the predicted Capture Point using the Linear Inverted Pendulum Model.

Learning resulted in a significant increase in the frequency of the robot coming to a stop and a significant decrease in the residual energy after a step in both the off-line learning case and the on-line learning case.

A. Off-line Learning

Figure 2 shows the results of stepping to various locations for a given push. Pink circles show Capture Points. At other points a red arrow shows the resulting Center of Mass position and a green arrow shows the resulting Center of Mass Velocity at the point where the Center of Mass comes closest to the center of the support ankle. The black and white target shows the ground projection of the Center of Mass when the Desired Step Location Calculator was queried. The blue rectangle shows the location of the current stance foot. The center of the grid of points is the location of the predicted Capture Point using the Linear Inverted Pendulum Model.

Figure 3 shows the measured and best fit stepping location offsets resulting from off-line learning. The magnitude and angle of the step location offset (vertical axes) are functions of the Center of Mass velocity angle at the decision event (horizontal axis). The offset magnitude is fit using a cubic curve. The angle is fit using a cubic curve for velocity angles less than 0, and a line for velocity angles greater than 0. Several points in each dimension were ignored as outliers when calculating the best fit curve, as the resulting fit passed near fewer of the measured points. These outliers imply that the Center of Mass velocity angle cannot completely characterize the offset vector of the Capture Point from the Linear Inverted Pendulum Capture Point. However, these curve fits provide the controller with a simple approximation that encompasses enough of the measured data to make a reasonable prediction.

Figure 4 shows the Stopping Error using Capture Points predicted from the Linear Inverted Pendulum Model and using a curve fit with off-line learning. Nine different force magnitudes from 150 to 550 Newtons for 0.05 seconds, corresponding to 7.5 to 27.5 $Nsec$ impulses were applied from 20 different directions. The mean squared error before learning is 0.137 $\frac{J^2}{Kg^2}$, and the controller successfully comes to a Stopped State only 1 out of 180 trials. After learning the mean squared error is 0.0192 $\frac{J^2}{Kg^2}$, and the controller successfully comes to a Stopped State in 45 out of 180 trials.

B. On-line Learning Results

Figure 5 shows the learning results of the on-line learning trials. Learning is turned on after 1000 trials and turned off after 3000 trials. We see that the pre-learning success rate is about 60% and the post-learning success rate is about 90%.

Figure 6 shows a representation of the Radial Basis Function Memory. The three rows of the graph show the memory throughout the training process, at 100, 200, and 1000 learning iterations. The horizontal and vertical axes of each plot are the x and y location of the Linear Inverted Pendulum Capture Point. The plots from left to right represent slices of the memory at progressively larger Center of Mass distances from the support foot. The Center of Mass range is represented as a black ring around the center of the foot, represented as a blue rectangle. The coloring in each plot represents the offset of the learned Capture Point from the Linear Inverted Pendulum Capture Point.

VI. Discussion and Future Work

We have presented a method for learning Capture Points for Humanoid Push Recovery. This method assumes that a Stepping Controller and a Stopping Controller already exist for the biped. We learn offsets to Capture Point locations predicted from the Linear Inverted Pendulum Model of walking. This results in a Desired Foot Placement Controller that the Stepping Controller can query. Simulation results show that this method results in significantly more robust push recovery than without using learning. There are several points of discussion and potential areas of future work.

A. What was Learned?

The intent of this study was to learn Capture Points for Humanoid Push Recovery. While we achieved improved push recovery performance through learning, it is interesting to think about what was really learned. While the Stepping Control System does a reasonable job of stepping to the desired step location, it does have some errors. Therefore, in addition to correcting for the deviation from the Linear Inverted Pendulum Model, we also adapt to the errors resulting from the Stepping Control System. In other words, we learned desired stepping locations that result in steps to actual Capture Points, even if the actual step is away from the desired step.

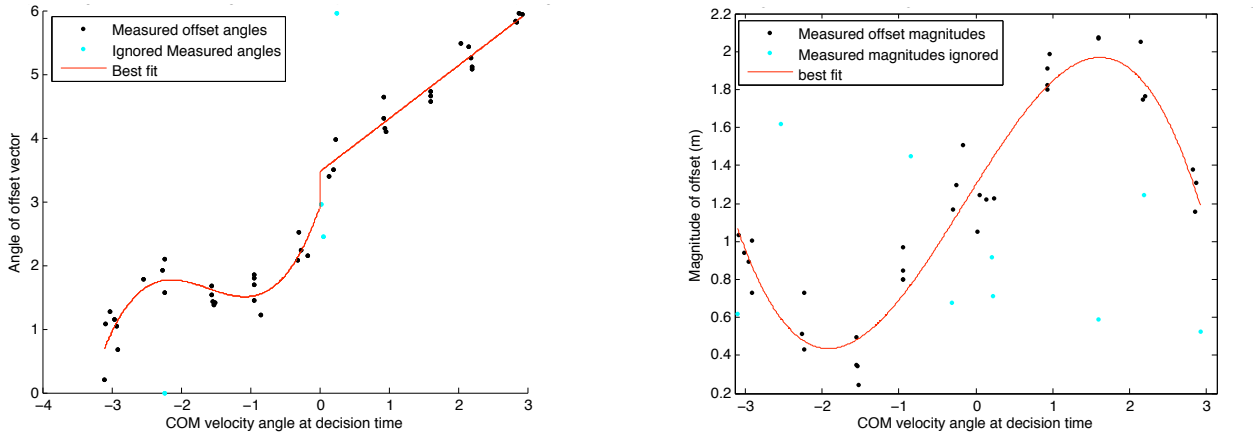

Fig. 3. Measured and best fit stepping location offsets resulting from off-line learning. The left graphs shows the angle of the step location offset (vertical axis) as a function of the Center of Mass velocity angle at the decision event (horizontal axis). The right graph shows the magnitude of the step location offset (vertical axis) as a function of the Center of Mass velocity angle at the decision event (horizontal axis).

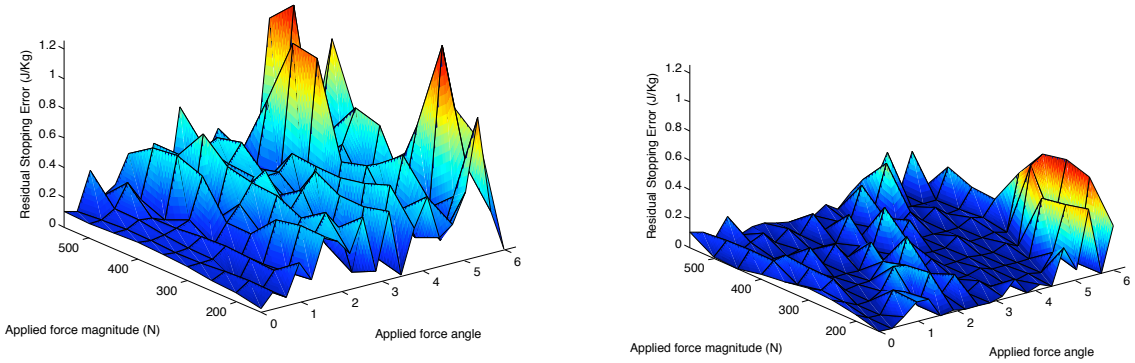

Fig. 4. Stopping Error using Capture Points predicted from the Linear Inverted Pendulum Model (left) and adding a Capture Point offset predicted from the simple curve fit based on the Center of Mass velocity angle at the decision event (right). Nine different force impulses were applied from 20 directions. The mean squared Stopping Error when using the Linear Inverted Pendulum model is 0.137 $\frac{J^2}{Kg^2}$, with a stopping success rate of 1 out of 180 trials. The mean squared error when using the curve fit to predict Capture Point offsets is 0.0192 $\frac{J^2}{Kg^2}$, with a success rate of 45 out of 180 trials.

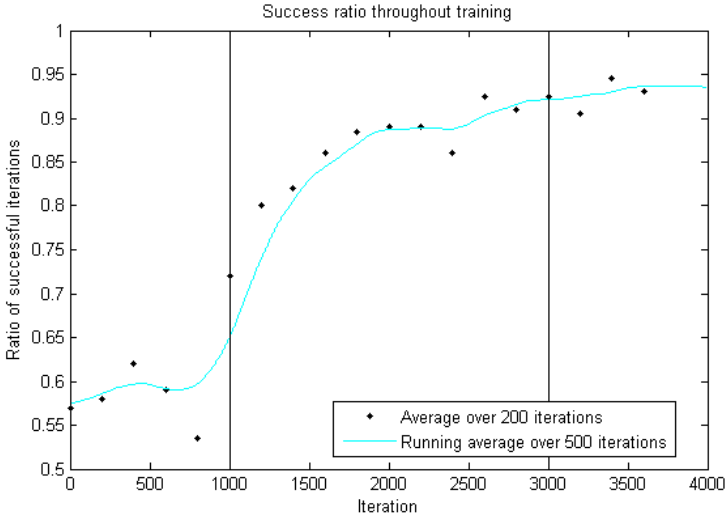

Fig. 5. Results of on-line learning. A successful trial has a value of 1, and a failed trial has a value of 0, regardless of the residual Energy Error. The points plotted represent the average value over 200 trials. The filtered estimate was found by using a forward/reverse 500 trial running average of the data padded on either side with the average of the first and last 500 iterations. The vertical lines represent training being turned on at 1000 iterations and off after 3000 iterations. The disturbance forces used to test the controller before and after training are identical.

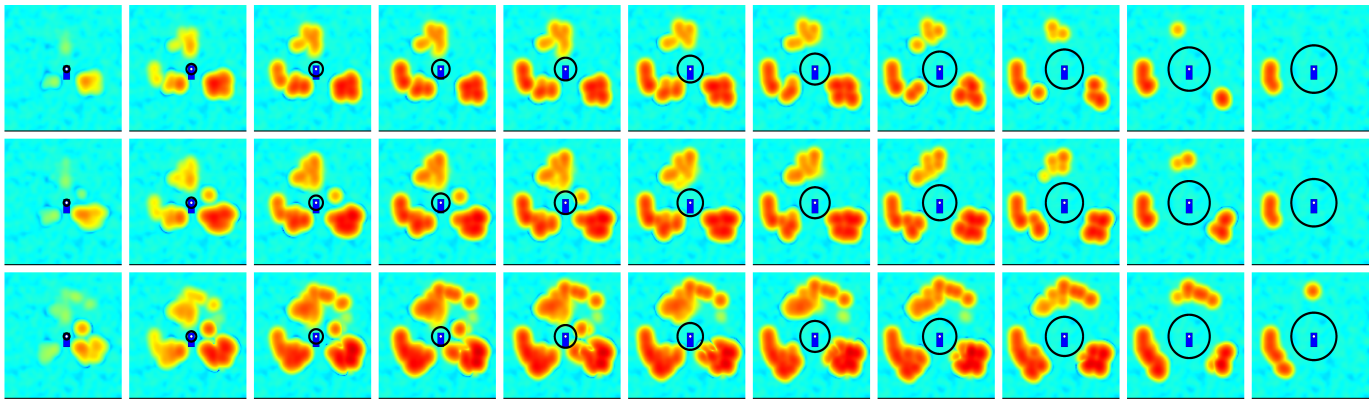

Fig. 6. A representation of the learned memory at 100 (top row), 200 (middle row), and 1000 (bottom row) iterations for on-line learning. The black ring represents the range of locations of the Center of Mass for the given graph. The blue rectangle represents the foot location. The XY space represents the Linear Inverted Pendulum Capture Point location. The color at a point represents the magnitude of the offset from the Linear Inverted Pendulum Capture Point to the corresponding Capture Point in memory. The scale is logarithmic such that blue regions represent offset magnitudes of less than 3 centimeters, dark red represents a 30 centimeter offset.

B. Generalizing to More States

In this study we limited the initial setup of the robot to start standing and balancing on one foot with the other foot slightly off the ground. Then a disturbance force was applied. By limiting the initial setup to these conditions, the effective size of the space that needed to be generalized over was relatively small. Since only the magnitude and direction of the disturbance push were varied between trials, only a two dimensional subspace of the full state space was visited. This limited exploration of state space allowed us to avoid the Curse of Dimensionality and easily learn Capture Point locations using standard function approximators and low dimensional Radial Basis Function Networks with gridded center spacing.

This limited approach is justifiable as a first step toward general push recovery since the learned Capture Point locations can be used as a baseline for a broader range of state space exploration. Similarly to a toddler learning to walk, the ability to recover from a push when standing can be built on to develop the ability to recover from pushes while walking.

We are now expanding our technique to learn to recover from pushes that occur at random times during walking. This effort may require more complex learning techniques that perform good generalization over a large range of state space. However, by starting with the controller that is learned using the techniques presented in this paper, learning for the more general case should be sped up.

C. Improving the Stopping Controller

In this study, we used a Stopping Controller that utilizes foot Center of Pressure regulation as the main strategy for stopping the robot and balancing over the foot. Our Stopping Controller did not use acceleration of internal angular momentum (lunging the body or windmilling the arms), or modification of the Center of Mass trajectory to assist in stopping. These two neglected strategies can be quite powerful both in expanding the range of pushes that a biped can recover from and in expanding the Capture Region for a given push.

We did not use these two strategies in this study simply because we do not have a good stopping controller that uses these strategies yet. We are currently developing such Stopping Controllers and have shown preliminary work for planar bipeds [26], [27]. Because we set up the problem formulation in this paper such that any reasonable stopping controller can be used, as our Stopping Controllers are improved, we should be able to use them in the current framework without requiring modification to the framework.

D. Learning Two-Step and Three-Step Capture Points

In this paper we learned One-Step Capture Points, i.e. points that could be stepped to in order to stop in one step. There are several reasons why we chose to stop in one step. From a practical point of view, there are many instances where a biped may need to stop in one step when footholds are limited and minimal body motion is allowable. For example, if crossing a cluttered room, there may only be one area to step to so the robot will get only one chance to stop. From a robustness point of view, stopping in one step is harder than stopping in multiple steps, so a controller that can stop or almost stop in one step after various pushes should be able to easily stop in two or three steps after these same pushes. For example, for each of the trials that our simulation did not stop in one step after learning, the Residual Stopping Error was very low and had the robot simply put its trailing foot down next to the new Support Foot, the robot likely would have recovered from the push.

In this paper, we did limit the magnitude of pushes to those in which the robot can stop in one step. For higher magnitude pushes, it may be theoretically impossible to stop in a single step, in which case two or more steps may be required, if the robot can recover at all. Therefore, it may be useful to also learn Two-Step, Three-Step, and perhaps even N-Step Capture Points. Intuitively, it seems that the best strategy for recovering from a large push in which a One-Step Capture Point is not reachable is to step as far as possible as fast as possible in the

direction of the Humanoid's Center of Mass velocity vector until, after a few steps are taken, a One-Step Capture Point is now reachable.

Knowing to what extent a Humanoid can stop in a given number of steps is a potentially useful stability and robustness margin [28]. Learning One-Step Capture Points allows us to also determine if a Humanoid is One-Step Capturable. We could probably extend the methods in this paper to also learn the One-Step Capture Margin, given a robot's state. If we were able to learn N-Step Capture Points and/or N-Step Capture Margins for large N, then we would have a way to predict whether a Humanoid is eventually stoppable, or will eventually fall.

E. Application to a Real Robot

We are currently building a Humanoid robot with inertial parameters that are close to the simulation model used for this paper. The robot should be operational in the Summer of 2008. The first experiments on this robot will be for Humanoid Push Recovery, including reproducing the results from this paper on the real robot.

REFERENCES

[1] Chee Meng Chew and G.A. Pratt. Frontal plane algorithms for dynamic bipedal walking. *Robotica*, 22(1):29–39, 2004.

[2] Chee Meng Chew and Gill A. Pratt. Dynamic bipedal walking assisted by learning. *Robotica*, 20(5):477–491, 2002.

[3] T. Fukuda, Y. Komata, and T. Arakawa. Stabilization control of biped locomotion robot based learning with gas having self-adaptive mutation and recurrent neural networks. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 217–222, 1997.

[4] K. Hitomi, T. Shibata, Y. Nakamura, and S. Ishii. On-line learning of a feedback controller for quasi-passive-dynamic walking by a stochastic policy gradient method. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3803–3808, 2005.

[5] Andreas Hofmann. A sliding controller for bipedal balancing using integrated movement of contact and non-contact limbs. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1952–1959, 2004.

[6] Andreas Hofmann. *Robust Execution of Bipedal Walking Tasks From Biomechanical Principles*. PhD thesis, Massachusetts Institute of Technology, 2006.

[7] Sang-Ho Hyon and Gordon Cheng. Gravity compensation and full-body balancing for humanoid robots. *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 214–221, 2006.

[8] Sang-Ho Hyon and Gordon Cheng. Disturbance rejection for biped humanoids. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2668–2675, 2007.

[9] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 239–246, 2001.

[10] Jae Won Kho, Dong Cheol Lim, and Tae Yong. Kuc. Implementation of an intelligent controller for biped walking using genetic algorithm. *Proceeings of the IEEE International Symposium on Industrial Electronics*, pages 49–54, 2006.

[11] Jaewon Kho and Dongcheol Lim. A learning controller for repetitive gait control of biped walking robot. *Proceedings of the SICE Annual Conference*, pages 885–889, 2004.

[12] T. Komura, H. Leung, S. Kudoh, and J. Kuffner. A feedback controller for biped humanoids that can counteract large perturbations during gait. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2001–2007, 2005.

[13] T.Y. Kuc, S.M. Baek, H.G. Lee, and J.O. Kim. Fourier series learning of biped walking motion. *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*, pages 49–54, 2002.

[14] Qinghua Li, Atsuo Takanishi, and Ichiro Kato. Learning control for a biped walking robot with a trunk. *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems.*, pages 1771–1777, 1993.

[15] Qinghua Li, Atsuo Takanishi, and Ichiro Kato. Learning of robot biped walking with the cooperation of human. *Proceedings of the IEEE International Workshop on Robot and Human Communication*, pages 393–397, 1993.

[16] Thijs Mandersloot, Martijn Wisse, and Christopher G. Atkeson. Controlling velocity in bipedal walking: A dynamic programming approach. *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 124–130, 2006.

[17] Yong Mao, Jiaxin Wang, Peifa Jia, Shi Li, Zhen Qiu, Le Zhang, and Zhuo Han. A reinforcement learning based dynamic walking control. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3609–3614, 2007.

[18] W.T. Miller. Learning dynamic balance of a biped walking robot. *Proceedings of the 1994 IEEE International Conference on Neural Networks.IEEE World Congress on Computational Intelligence*, pages 2771–2776, 1994.

[19] T. Mori, Y. Nakamura, M. Sato, and S. Ishii. Reinforcement learning for a cpg-driven biped robot. *Proceedings of the Nineteenth National Conference on Artificial Intelligence, AAAI*, 2004.

[20] J. Morimoto, G. Cheng, G. Atkeson, and G. Zeglin. A simple reinforcement learning algorithm for biped locomotion. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3030–3035, 2004.

[21] J. Morimoto, J. Nakanishi, G. Endo, G. Cheng, C.G. Atkeson, and G. Zeglin. Poincare-map-based reinforcement learning for biped walking. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2381–2386, 2005.

[22] J. Morimoto, G. Zeglin, and C.G. Atkeson. Minimax differential dynamic programming: application to a biped walking robot. *SICE 2003 Annual Conference*, pages 2310–2315, 2003.

[23] Jun Morimoto and Christopher G. Atkeson. Learning biped locomotion: Application of poincare-map-based reinforcement learning. *IEEE Robotics and Automation Magazine*, pages 41–51, 2007.

[24] Masaki Ogino, Yutaka Katoh, Masahiro Aono, Minoru Asada, and Koh Hosoda. Vision-based reinforcement learning for humanoid behavior generation with rhythmic walking parameters. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1665–1671, 2003.

[25] Jerry E. Pratt. *Exploiting Inherent Robustness and Natural Dynamics in the Control of Bipedal Walking Robots*. PhD thesis, Massachussetts Institute of Technology, May 2000.

[26] Jerry E. Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 200–207, 2006.

[27] Jerry E. Pratt and Sergey V. Drakunov. Derivation and application of a conserved orbital energy for the inverted pendulum bipedal walking model. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4653–4660, 2007.

[28] Jerry E. Pratt and Russ Tedrake. Velocity based stability margins for fast bipedal walking. *In Fast Motions in Robotics and Biomechanics - Optimization and Feedback Control.M.Diehl, K.D.Mombaur (editors), Lecture Notes in Information Science, Springer, No.340, Sept.2006.*, 2006.

[29] E. Schuitema, D.G.E. Hobbelen, P.P. Jonker, M. Wisse, and J.G.D. Karssen. Using a controller based on reinforcement learning for a passive dynamic walking robot. *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 232–237, 2005.

[30] Russ Tedrake, Teresa Weirui Zhang, and H.Sebastian Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 2849–2854, 2004.

[31] Shouyi Wang, Jelmer Braaksma, Robert Babuska, and Daan Hobbelen. Reinforcement learning control for biped robot walking on uneven surfaces. *Proceedings of the 2006 International Joint Conference on Neural Networks*, pages 4173–4178, 2006.

[32] Changjiu Zhou and Qingchun Meng. Reinforcement learning with fuzzy evaluative feedback for a biped robot. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3829–3834, 2000.