Stochastic Policy Gradient Reinforcement Learning on a Simple 3D Biped

Russ Tedrake
Computer Science &
Artificial Intelligence Lab,
Center for Bits & Atoms
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: russt@ai.mit.edu

Teresa Weirui Zhang
Department of
Mechanical Engineering
Brain & Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: resa@mit.edu

H. Sebastian Seung
Howard Hughes Medical Institute
Brain & Cognitive Sciences
Center for Bits & Atoms
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: seung@mit.edu

***Abstract*— We present a learning system which is able to quickly and reliably acquire a robust feedback control policy for 3D dynamic walking from a blank-slate using only trials implemented on our physical robot. The robot begins walking within a minute and learning converges in approximately 20 minutes. This success can be attributed to the mechanics of our robot, which are modeled after a passive dynamic walker, and to a dramatic reduction in the dimensionality of the learning problem. We reduce the dimensionality by designing a robot with only 6 internal degrees of freedom and 4 actuators, by decomposing the control system in the frontal and sagittal planes, and by formulating the learning problem on the discrete return map dynamics. We apply a stochastic policy gradient algorithm to this reduced problem and decrease the variance of the update using a state-based estimate of the expected cost. This optimized learning system works quickly enough that the robot is able to continually adapt to the terrain as it walks.**

I. Introduction

Recent advances in bipedal walking technology have produced robots capable of leaving the laboratory environment to interact with the unknown and uncertain environments of the real world. Despite our best efforts, it is unlikely that we will be able to preprogram these robots for every possible situation without sacrificing performance. Endowing our robots with the ability to learn from experience and adapt to their environment seems critical for the success of any real world robot.

Dynamic bipedal walking is difficult to learn for a number of reasons. First, walking robots typically have many degrees of freedom, which can cause a combinatorial explosion for learning systems that attempt to optimize performance in every possible configuration of the robot. Second, details of the robot dynamics such as uncertainties in the ground contact and nonlinear friction in the joints are difficult to model well in simulation, making it unlikely that a controller optimized in a simulation will perform optimally on the real robot. Since it is only practical to run a small number of learning trials on the real robot, the learning algorithms must perform well after obtaining a very limited amount of data. Finally, learning algorithms for dynamic walking must deal with dynamic discontinuities caused by collisions with the ground and with the problem of delayed reward - torques applied at one time may have an effect on the performance many steps into the future.

Although there is a great deal of literature on learning control for dynamically walking bipedal robots, there are relatively few examples of learning algorithms actually implemented on the robot or which work quickly enough to allow the robot to adapt online to changing terrain. Some researchers attempt to learn a controller in simulation that is robust enough to run on the real robot [1], treating differences between the simulation and the robot as disturbances. The University of New Hampshire bipeds were two early examples of online learning which acquired a basic gait by tuning parameters in a hand-designed controller ([2], [3]). In this paper we generalize these results to obtaining a controller from scratch instead of tuning an existing controller. Learning control has also been successfully implemented on Sony's quadrupedal robot AIBO (i.e., [4]). The learned controllers for AIBO are open-loop trajectories, but trajectory feedback is essential for robust, dynamic, bipedal walking.

In order to study learning feedback control for walking, we performed our initial experiments on a simplified robot which captures the essence of dynamic walking but which minimizes many of the complications. Our robot has only 6 internal degrees of freedom and 4 actuators[1]. The mechanical design of our robot is based on a passive dynamic walker ([5], [6]). This allows us to solve a portion of the control problem in the mechanical design, and makes the robot mechanically very stable; most policies in our search space result in either stable walking or failed walking where the robot ends up simply standing still.

The learning on our robot is performed by a policy gradient reinforcement learning algorithm ([7], [8], [9]). The goal of this paper is to describe our formulation of the learning problem and the algorithm that we use to solve it. We include our experimental results on this simplified biped, and discuss the possibility of applying the same algorithm to a more complicated walking system.

[1]The standard for 3D bipeds is to have at least 12 internal degrees of freedom and 12 actuators in the legs

II. The Robot

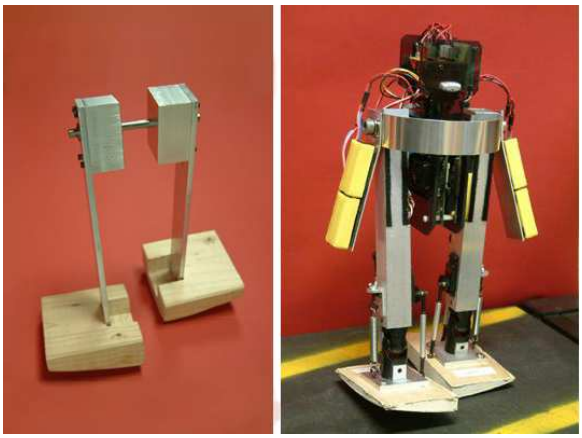

Fig. 1. The robot on the left is a simple passive dynamic walker. The robot on the right is our actuated version of the same robot.

The passive dynamic walker shown on the left in Figure 1 represents the simplest machine that we could build which captures the essence of stable dynamic walking in three dimensions. It has only a single passive pin joint at the hip. When placed at the top of a small ramp and given a push sideways, the walker will begin falling down the ramp and eventually converge to a stable limit cycle trajectory that has been compared to the waddling gait of a penguin [10]. The energetics of this passive walker are common to all passive walkers: energy lost due to friction and collisions when the swing leg returns to the ground is balanced by the gradual conversion of potential energy into kinetic energy as the walker moves down the slope. The mechanical design of this robot and some experimental stability results are presented in [11].

We designed our learning robot by adding a small number of actuators to this passive design. The robot shown on the right in figure 1, which is also described in [11], has passive joints at the hip and 2 degrees of actuation (roll and pitch) at each ankle. The ankle actuators are position controlled servo motors which, when commanded to hold their zero position, allow the actuated robot to walk stably down a small ramp, "simulating" the passive walker. The shape of the large, curved feet is designed to make the robot walk passively at 0.8Hz, and to take steps of approximately 6.5 cm when walking down a ramp of 0.03 radians. The robot stands 44 cm tall and weighs approximately 2.9 kg, which includes the CPU and batteries that are carried on-board. The most recent additions to this robot are the passive arms, which are mechanically coupled to the opposite leg to provide mechanical yaw compensation.

When placed on flat terrain, the passive walker waddles back and forth, slowly losing energy, until it comes to rest standing still. In order to achieve stable walking on flat terrain, the actuators on our learning robot must restore energy into the system that would have been restored by gravity when walking down a slope.

III. The Learning Problem

The goal of learning is to acquire a feedback control policy which makes the robot's gait invariant to small slopes. In total, the system has 9 degrees of freedom[2], and the equations of motion can be written in the form

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau + \mathbf{D}(t), \tag{1}$$

where

$$\begin{aligned}\mathbf{q} =&[\theta_{yaw}, \theta_{lPitch}, \theta_{bPitch}, \theta_{rPitch}, \theta_{roll},\\ &\theta_{raRoll}, \theta_{laRoll}, \theta_{raPitch}, \theta_{laPitch}]^T,\\ \tau =&[0,0,0,0,\tau_{raRoll}, \tau_{laRoll}, \tau_{raPitch}, \tau_{laPitch}]^T.\end{aligned}$$

$\mathbf{H}$ is the state dependent inertial matrix, $\mathbf{C}$ contains interaction torques between the links, $\mathbf{G}$ represents the effect of gravity, τ are the motor torques, and $\mathbf{D}$ are random disturbances to the system. Our shorthand $lPitch$, $bPitch$, and $rPitch$ refer to left leg pitch, body pitch, and right leg pitch, respectively. $raRoll$, $laRoll$, $raPitch$, and $laPitch$ are short for right and left ankle roll and pitch. The actual output of the controller is a motor command vector

$$\mathbf{u} = [u_{raRoll}, u_{laRoll}, u_{raPitch}, u_{laPitch}]^T,$$

which generates torques

$$\tau = h(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}).$$

The function h describes the linear feedback controller implemented by the servo boards and the nonlinear kinematic transformation into joint torques.

The robot uses a deterministic feedback control policy which is represented using a linear function approximator parameterized by vector $\mathbf{w}$ and using nonlinear features ϕ:

$$\mathbf{u} = \pi_{\mathbf{w}}(\hat{\mathbf{x}}) = \sum_i w_i \phi_i(\hat{\mathbf{x}}), \text{ with } \mathbf{x} = \begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix}. \tag{2}$$

The notation $\hat{\mathbf{x}}$ represents a noisy estimate of the state $\mathbf{x}$. Before learning, $\mathbf{w}$ is initialized to all zeros, making the policy outputs zero everywhere, so that the robot simulates the passive walker.

To quantify the stability of our nonlinear, stochastic, periodic trajectory, we consider the dynamics on the return map, taken around the point where $\theta_{roll} = 0$ and $\dot{\theta}_{roll} > 0$. The return map dynamics are a Markov random sequence with the probability at the $(n+1)$th crossing of the return map given by

$$f_{\mathbf{w}}(\mathbf{x}', \mathbf{x}) = P\{\hat{\mathbf{X}}(n+1) = \mathbf{x}' | \hat{\mathbf{X}}(n) = \mathbf{x}, \mathbf{W}(n) = \mathbf{w}\}. \tag{3}$$

$f_{\mathbf{w}}(\mathbf{x}', \mathbf{x})$ represents the probability density function over the state space which contains the dynamics in equations 1 and 2 integrated over one cycle. We do not make any assumptions about its form, except that it is Markov. Note that the element of $f_{\mathbf{w}}$ representing θ_{roll} is the delta function, independent of

[2]6 internal DOFs and 3 DOFs for the robot's orientation. We assume that the robot is always in contact with the ground at a single point, and infer the robot's absolute (x, y) position in space directly from the remaining variables.

$\mathbf{x}$. The return map dynamics are represented as a Markov chain that depends on the parameter vector $\mathbf{w}$ instead of the equivalent Markov decision process for simplification because the feedback controller is evaluated many times during a single step (our controller runs at 100 Hz and our robot steps at around 0.8 Hz). The stochasticity in $f_{\mathbf{w}}$ comes from the random disturbances $\mathbf{D}(t)$ and the state estimation error, $\hat{\mathbf{x}}-\mathbf{x}$.

The cost function for learning uses a constant desired value, $\mathbf{x}^d$, on the return map:

$$g(\mathbf{x}) = \frac{1}{2}|\mathbf{x} - \mathbf{x}^d|^2. \tag{4}$$

This desired value can be considered a reference trajectory on the return map, and is taken from the gait of the walker down a slope of 0.03 radians; no reference trajectory is required for the limit cycle between steps. For a given trajectory $\hat{\mathrm{x}} = [\hat{\mathbf{x}}(0), \hat{\mathbf{x}}(1), ..., \hat{\mathbf{x}}(N)]$, we define the average cost

$$G(\hat{\mathrm{x}}) = \frac{1}{N}\sum_{n=0}^{N} g(\hat{\mathbf{x}}(n)). \tag{5}$$

Our goal is to find the parameter vector $\mathbf{w}$ which minimizes

$$\lim_{N\to\infty} E\left\{G(\hat{\mathrm{x}})\right\}. \tag{6}$$

By minimizing this error, we are effectively minimizing the eigenvalues of return map, and maximizing the stability of the desired limit cycle.

IV. The Learning Algorithm

The learning algorithm is a statistical algorithm which makes small changes to the control parameters $\mathbf{w}$ on each step and uses correlations between changes in $\mathbf{w}$ and changes in the return map error to climb the performance gradient. This can be accomplished with a very simple online learning rule which changes $\mathbf{w}$ with each step that the robot takes. The particular algorithm that we present here was originally proposed by [7]. We present a thorough derivation of this algorithm in the next section.

The algorithm makes use of an intermediate representation which we call the *value function*, $J(\mathbf{x})$. The value of state $\mathbf{x}$ is the expected average cost to be incurred by following policy $\pi_{\mathbf{w}}$ starting from state $\mathbf{x}$:

$$J(\mathbf{x}) = \lim_{N\to\infty}\frac{1}{N}\sum_{n=0}^{N} g(\mathbf{x}(n)), \text{ with } \mathbf{x}(0) = \mathbf{x}.$$

$\hat{J}_{\mathbf{v}}(\mathbf{x})$ is an estimate of the value function parameterized by vector $\mathbf{v}$. This value estimate is represented in another function approximator:

$$\hat{J}_{\mathbf{v}}(\hat{\mathbf{x}}) = \sum_i v_i\psi_i(\hat{\mathbf{x}}). \tag{7}$$

During learning, we add stochasticity to our deterministic control policy by varying $\mathbf{w}$. Let $\mathbf{Z}(n)$ be a Gaussian random vector with $E\{Z_i(n)\} = 0$ and $E\{Z_i(n)Z_j(n')\} = \sigma^2\delta_{ij}\delta_{nn'}$. During the nth step that the robot takes, we evaluate the controller using the parameter vector $\mathbf{w}'(n) = \mathbf{w}(n) + \mathbf{z}(n)$. The algorithm uses a storage variable, $\mathbf{e}(n)$, which we call the eligibility trace. We begin with $\mathbf{w}(0) = \mathbf{e}(0) = \mathbf{0}$. At the end of the nth step, we make the updates:

$$\delta(n) = g\left(\hat{\mathbf{x}}(n)\right) + \gamma\hat{J}_{\mathbf{v}}\left(\hat{\mathbf{x}}(n+1)\right) - \hat{J}_{\mathbf{v}}\left(\hat{\mathbf{x}}(n)\right) \tag{8}$$

$$e_i(n) = \gamma e_i(n-1) + b_i(n)z_i(n) \tag{9}$$

$$\Delta w_i(n) = -\eta_w\delta(n)e_i(n) \tag{10}$$

$$\Delta v_i(n) = \eta_v\delta(n)\psi_i(\hat{\mathbf{x}}(n)). \tag{11}$$

$\eta_w \geq 0$ and $\eta_v \geq 0$ are the learning rates and γ is the discount factor of the eligibility trace, which will be discussed in more detail in the algorithm derivation. $b_i(n)$ is a boolean one step eligibility, which is 1 if the parameter w_i is activated ($\phi_i(\hat{\mathbf{x}}) > 0$) at any point during step n and 0 otherwise. $\delta(n)$ is called the one step *temporal difference error*.

The algorithm can be understood intuitively. On each step the robot receives some cost $g(\hat{\mathbf{x}}(n))$. This cost is compared to cost that we expect to receive, as estimated by $\hat{J}_{\mathbf{v}}(\mathbf{x})$. If the cost is lower than expected, then $-\eta\delta(n)$ is positive, so we add a scaled version of the noise terms, z_i, into w_i. Similarly, if the cost is higher than expected, then we move in the opposite direction. This simple online algorithm performs approximate stochastic gradient descent on the expected value of the average infinite-horizon cost.

V. Algorithm Derivation

The expected value of the average cost, G, is given by:

$$E\{G(\hat{\mathrm{x}})\} = \int_{\hat{\mathrm{x}}} G(\hat{\mathrm{x}})P_{\mathbf{w}'}\{\hat{\mathbf{X}} = \hat{\mathrm{x}}\}d\hat{\mathrm{x}}.$$

The probability of trajectory $\hat{\mathrm{x}}$ is

$$P_{\mathbf{w}'}\{\hat{\mathbf{X}} = \hat{\mathrm{x}}\} = P\{\hat{\mathbf{X}}(0) = \hat{\mathbf{x}}(0)\}\prod_{n=0}^{N-1} f_{\mathbf{w}'}(\hat{\mathbf{x}}(n+1), \hat{\mathbf{x}}(n)).$$

Taking the gradient of $E\{G(\hat{\mathrm{x}})\}$ with respect to $\mathbf{w}$ we find

$$\begin{aligned}
\frac{\partial}{\partial w_i}E\{G(\hat{\mathrm{x}})\} &= \int_{\hat{\mathrm{x}}} G(\hat{\mathrm{x}})\frac{\partial}{\partial w_i}P_{\mathbf{w}'}\{\hat{\mathbf{X}} = \hat{\mathrm{x}}\}d\hat{\mathrm{x}} \\
&= \int_{\hat{\mathrm{x}}} G(\hat{\mathrm{x}})P_{\mathbf{w}'}\{\hat{\mathbf{X}} = \hat{\mathrm{x}}\}\frac{\partial}{\partial w_i}\log P_{\mathbf{w}'}\{\hat{\mathbf{X}} = \hat{\mathrm{x}}\}d\hat{\mathrm{x}} \\
&= E\left\{G(\hat{\mathrm{x}})\frac{\partial}{\partial w_i}\log P_{\mathbf{w}'}\{\hat{\mathbf{X}} = \hat{\mathrm{x}}\}\right\} \\
&= E\left\{G(\hat{\mathrm{x}})\left(\sum_{m=0}^{N-1}\frac{\partial}{\partial w_i}\log f_{\mathbf{w}'}(\hat{\mathbf{x}}(m+1), \hat{\mathbf{x}}(m))\right)\right\}
\end{aligned}$$

Recall that $f_{\mathbf{w}'}(\mathbf{x}', \mathbf{x})$ is a complicated function which includes the integrated dynamics of the controller and the robot. Nevertheless, $\frac{\partial}{\partial w_i}\log f_{\mathbf{w}'}$ is simply:

$$\begin{aligned}
&\frac{\partial}{\partial w_i}\log f_{\mathbf{w}'}(\mathbf{x}'(m+1), \mathbf{x}(m)) = \\
&\quad\frac{\partial}{\partial w_i}\log\left[P\{\hat{\mathbf{X}}' = \mathbf{x}'|\hat{\mathbf{X}} = \mathbf{x}, \mathbf{W}' = \mathbf{w}'\}P_{\mathbf{w}}\{\mathbf{W}' = \mathbf{w}'\}\right] \\
&= \frac{\partial}{\partial w_i}\log P_{\mathbf{w}}\{\mathbf{W}'(m) = \mathbf{w}'\} = \frac{z_i(m)}{\sigma^2}
\end{aligned}$$

Substituting, we have

$$\frac{\partial}{\partial w_i}E\{G(\hat{\mathbf{x}})\} = \frac{1}{N\sigma^2}E\left\{\left(\sum_{n=1}^{N} g(\hat{\mathbf{x}}(n))\right)\left(\sum_{m=0}^{N-1} z_i(m)\right)\right\}$$
$$= \frac{1}{N\sigma^2}E\left\{\sum_{n=0}^{N} g(\hat{\mathbf{x}}(n))\sum_{m=0}^{n} b_i(m)z_i(m)\right\}.$$

This final reduction is based on the observation that $E\{g(\hat{\mathbf{x}}(n))\sum_{m=n}^{N} z_i(m)\} = 0$ (noise added to the controller on or after step n is not correlated to the cost at step n). Similarly, random changes to a weight that is not used during the nth step ($b_i(m) = 0$) have zero expectation, and can be excluded from the sum.

Observe that the variance of this gradient estimate grows without bound as $N \to \infty$ [8]. To bound the variance, we use a *biased* estimate of this gradient which artificially discounts the eligibility trace:

$$\frac{\partial}{\partial w_i}E\{G(\hat{\mathbf{x}})\} \approx$$
$$\frac{1}{N\sigma^2}E\left\{\sum_{n=0}^{N} g(\hat{\mathbf{x}}(n))\sum_{m=0}^{n} \gamma^{n-m} b_i(m)z_i(m)\right\}$$
$$= \frac{1}{N\sigma^2}E\left\{\sum_{n=0}^{N} g(\hat{\mathbf{x}}(n))e_i(n)\right\},$$

with $0 \leq \gamma \leq 1$. The discount factor γ parameterizes the bias-variance trade-off.

Next, observe that we can subtract any mean zero baseline from this quantity without effecting the expected value of our estimate [12]. Including this baseline can dramatically improve the performance of our algorithm because it can reduce the variance of our gradient estimate. In particular, we subtract a mean-zero term containing an estimate of the value function as recommended by [7]:

$$\lim_{N\to\infty}\frac{\partial}{\partial w_i}E\{G(\hat{\mathbf{x}})\} \approx$$
$$\lim_{N\to\infty}\frac{1}{N}E\left\{\sum_{n=0}^{N} g(\mathbf{x}(n))e_i(n) - \sum_{n=0}^{N}\hat{V}^{\pi}(\mathbf{x}(n))z_i(n)\right\}$$
$$= \lim_{N\to\infty}\frac{1}{N}E\left\{\sum_{n=0}^{N} g(\mathbf{x}(n))e_i(n) + \right.$$
$$\left.\sum_{n=0}^{N} z_i(n)\sum_{m=n}^{N-1}\gamma^{m-n}\left(\gamma\hat{V}^{\pi}(\mathbf{x}(m+1)) - \hat{V}^{\pi}(\mathbf{x}(m))\right)\right\}$$
$$= \lim_{N\to\infty}\frac{1}{N}E\left\{\sum_{n=0}^{N} g(\mathbf{x}(n))e_i(n) + \right.$$
$$\left.\sum_{n=0}^{N}\left(\gamma\hat{V}^{\pi}(\mathbf{x}(n+1)) - \hat{V}^{\pi}(\mathbf{x}(n))\right)e_i(n)\right\}$$
$$= \lim_{N\to\infty}\frac{1}{N}E\left\{\sum_{n=0}^{N}\delta(n)e_i(n)\right\}$$

By this derivation, we can see that the average of the weight update given in equations 8-11 is in the direction of the performance gradient:

$$\lim_{N\to\infty}\frac{1}{N}E\left\{\sum_{n=0}^{N}\Delta w_i(n)\right\} \approx -\eta\lim_{N\to\infty}\frac{\partial}{\partial w_i}E\{G(\hat{\mathbf{x}})\}.$$

VI. Learning Implementation

In our initial implementation of the algorithm, we decided to further simplify the problem by decomposing the control in the frontal and sagittal planes. In this decomposition, the ankle roll actuators are responsible for stabilizing the oscillations of the robot in the frontal plane. The ankle pitch actuators cause the robot to lean forward or backward, which moves the position of the center of mass relative to the ground contact point on the foot. Because the hip joint on our robot is passive, if the center of mass is in front of the ground contact when the swing foot leaves the ground, then the robot will begin to walk forward. The distance of between the center of mass and the ground contact is monotonically related to the step size and to the walking speed.

Due to the simplicity of the sagittal plane control, we only need to learn a control policy for the two ankle roll actuators which stabilize the roll oscillation in the frontal plane. This strategy will change as the robot walks at different speeds, but we hope the learning algorithm will adapt quickly enough to compensate for those differences.

With these simplifications in mind, we constrain the feedback policy to be a function of only two variables: θ_{roll} and $\dot{\theta}_{roll}$. The choice of these two variables is not arbitrary; they are the only variables that we use when writing a non-learning feedback controller that stabilizes the oscillation. We also constrain the policy to be symmetric - the controller for the left ankle is simply a mirror image of the controller for the right ankle. Therefore, the learned control policy only has a single output. The value function is approximated as a function of only a single variable: $\dot{\theta}_{roll}$. This very low dimensionality allows the algorithm to train very quickly.

The control policy and value functions are both represented using linear function approximators of the form described in Equations 2 and 7, which are fast and very convenient to initialize. We use a non-overlapping tile-coding for our approximator basis functions: 35 tiles for the policy (5 in $\theta_{roll}\times$ 7 in $\dot{\theta}_{roll}$) and 11 tiles for the value function.

In order to make the robot explore the state space during learning, we hand-designed a simple controller to place the robot in random initial conditions on the return map. The random distribution is biased according to the distribution of points that the robot has already experienced on the return map - the most likely initial condition is the state that the robot experienced least often. We use this controller to randomly reinitialize the robot every time that it comes to a halt standing still, or every 10 seconds, whichever comes first. This heuristic makes the distribution on the return map more uniform, and increases the likelihood of the algorithm converging on the same policy each time that it learns from a blank slate.

VII. Experimental Results

When the learning begins, the policy parameters, $\mathbf{w}$, are set to $\mathbf{0}$ and the baseline parameters, $\mathbf{v}$, are initialized so that $\hat{J}_{\mathbf{v}}(\mathbf{x}) \approx \frac{g(\mathbf{x})}{1-\gamma}$. We typically train the robot on flat terrain using short trials with random initial conditions. During the first few trials, the policy does not restore sufficient energy into the system, and the robot comes to a halt standing still. Within a minute of training, the robot achieves foot clearance on nearly every step; this is the minimal definition of walking on this system. The learning easily converges to a robust gait with the desired fixed point on the return map within 20 minutes (approximately 960 steps at 0.8 Hz). Error obtained during learning depends on the random initial conditions of each trial, and is therefore a very noisy stochastic variable. For this reason, in Figure 2 we plot a typical learning curve in terms of the *average* error per step. Figure 3 plots a typical trajectory of the learned controller walking on flat terrain. Figure 4 displays the final policy.

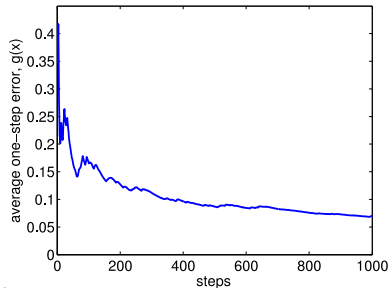

Fig. 2. A typical learning curve, plotted as the average error on each step.

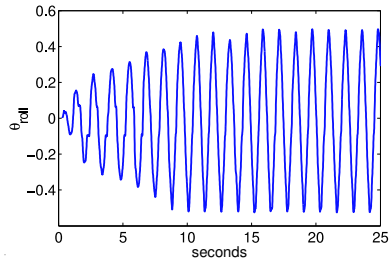

Fig. 3. θ_{roll} trajectory of the robot starting from standing still.

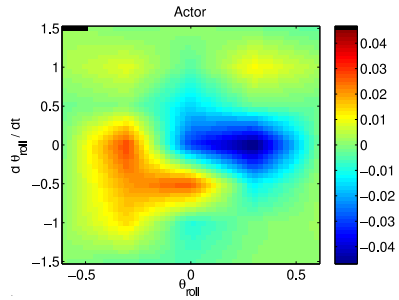

Fig. 4. Learned feedback control policy $u_{raRoll} = \pi_{\mathbf{w}}(\hat{\mathbf{x}})$.

In Figure 5 we plot the return maps of the system before learning ($\mathbf{w} = \mathbf{0}$) and after 1000 steps. In general, the return map for our 9 DOF robot is 17 dimensional (9 states + 9 derivatives - 1), and the projection of these dynamics onto a single dimension is difficult to interpret. The plots in Figure 5 where made with the robot walking in place on flat terrain. In this particular situation, most of the return map variables are close to zero throughout the dynamics, and a two dimensional return map captures the desired dynamics. As expected, before learning the return map illustrates a single fixed point at $\dot{\theta}_{roll} = 0$, which means the robot is standing still. After learning, we obtain a single fixed point at the desired value ($\dot{\theta}_{roll} = 1.0$ radians / second), and the basin of attraction of this fixed point extends over the entire domain that we tested. On the rare occasion that the robot falls over, the system does not return to the map and stops producing points on this graph.

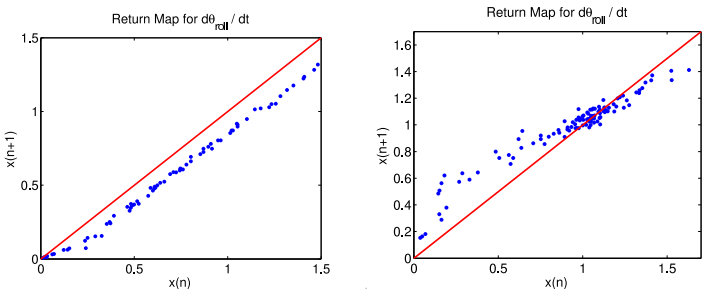

Fig. 5. Experimental return maps, before (left) and after (right) learning. Fixed points exist at the intersections of the return map (blue) and the line of slope one (red).

To quantify the stability of the learned controller, we measure the eigenvalues of the return map. Linearizing around the fixed point in Figure 5 suggests that the system has a single eigenvalue of 0.5. To obtain the eigenvalues of the return map when the robot is walking, we run the robot from a large number of initial conditions and record the return map trajectories $\hat{\mathbf{x}}^i(n)$, 9×1 vectors which represent the state of the system (with fixed ankles) on the nth crossing of the ith trial. For each trial we estimate $\hat{\mathbf{x}}^i(\infty)$, the equilibrium of the return map. Finally, we perform a least squares fit of the matrix $\mathbf{A}$ to satisfy the relation

$$[\hat{\mathbf{x}}^i(n+1) - \hat{\mathbf{x}}^i(\infty)] = \mathbf{A}[\hat{\mathbf{x}}^i(n) - \hat{\mathbf{x}}^i(\infty)].$$

The eigenvalues of $\mathbf{A}$ for the learned controller and for our hand-designed controllers (described in [11]) are:

Controller	Eigenvalues
Passive walking (63 trials)	$0.88 \pm 0.01i$, 0.75, $0.66 \pm 0.03i$, 0.54, 0.36, $0.32 \pm 0.13i$
Hand-designed feed-forward (89 trials)	0.80, 0.60, $0.49 \pm 0.04i$, 0.36, 0.25, $0.20 \pm 0.01i$, 0.01
Hand-designed feedback (58 trials)	0.78, $0.69 \pm 0.03i$, 0.36, 0.25, $0.20 \pm 0.01i$, 0.01
Learned feedback (42 trials)	$0.74 \pm 0.05i$, $0.53 \pm 0.09i$, 0.43, $0.30 \pm 0.02i$, 0.15, 0.07

All of these experiments were on flat terrain except the passive walking, which was on a slope of 0.027 radians. The

convergence of the system to the nominal trajectory is largely governed by the largest eigenvalues. This analysis suggests that our learned controller converges to the steady state trajectory more quickly that the passive walker on a ramp and more quickly than any of our hand-designed controllers.

Our stochastic policy gradient algorithm solves the temporal credit assignment problem in by accumulating the eligibility within a step and discounting eligibility between steps. Interestly, our algorithm performs best with heavy discounting between steps ($0 \leq \gamma \leq 0.2$). This suggests that our one dimensional value estimate does a good job of isolating the credit assignment to a single step.

While it took a few minutes to learn a controller from a blank slate, adjusting the learned controller to adapt to small changes in the terrain appears to happen very quickly. The non-learning controllers require constant attention and small manual changes to the parameters as the robot walks down the hall, on tiles, and on carpet. The learning controller easily adapts to these situations.

VIII. Discussion

Designing our robot like a passive dynamic walker changes the learning problem in a number of ways. It allows us to learn a policy with only a single output which controlled a 9 DOF system, and allows us to formulate the problem on the return map dynamics. It also dramatically increases the number of policies in the search space which could generate stable walking. The learning algorithm works extremely well on this simple robot, but will the technique scale to more complicated robots?

One factor in our success was the formulation of the learning problem on the discrete dynamics of the return map instead of the continuous dynamics along the entire trajectory. This formulation relies on the fact that our passive walker produces periodic trajectories even before the learning begins. It is possible for passive walkers to have knees and arms [13], or on a more traditional humanoid robot this algorithm could be used to augment and improve and existing walking controller which produces nominal walking trajectories.

As the number of degrees of freedom increases, the stochastic policy gradient algorithm may have problems with scaling. The algorithm correlates changes in the policy parameters with changes in the performance on the return map. As we add degrees of freedom, the assignment of credit to a particular actuator will become more difficult, requiring more learning trials to obtain a good estimate of the correlation. This scaling problem is an open and interesting research question and a primary focus of our current research.

IX. Conclusions

We have presented a learning formulation and learning algorithm which works very well on our simplified 3D dynamic biped. The robot begins to walk after only one minute of learning from a blank slate, and the learning converges to the desired trajectory in less than 20 minutes. This learned controller is quantifiably more stable, using the eigenvalues of the return map, than any controller we were able to derive for the same robot by hand. Once the controller is learned, to robot is able to quickly adapt to small changes in the terrain.

Building a robot to simplify the learning allowed us to gain some practical insights into the learning problem for dynamic bipedal locomotion. Implementing these algorithms on the real robot proved to be a very different problem than working in simulation. We would like to take two basic directions to continue this research. First, we are removing many of the simplifying assumptions used in this paper (such as the decomposed control policy) to better approximate optimal walking on this simple platform and to test our learning controller's ability to compensate for rough terrain. Second, we are scaling these results up to more sophisticated bipeds, including a passive dynamic walker with knees and humanoids that already have a basic control system in place.

Acknowledgments

This work was supported by the David and Lucille Packard Foundation (contract 99-1471), the National Science Foundation (grant CCR-0122419). Special thanks to Ming-fai Fong and Derrick Tan for their help with designing and building the experimental platform.

References

[1] J. Morimoto and C. Atkeson, "Minimax differential dynamic programming: An application to robust biped walking." Neural Information Processing Systems, 2002.

[2] W. T. Miller, III, "Real-time neural network control of a biped walking robot," *IEEE Control Systems Magazine*, vol. 14, no. 1, pp. 41–48, Feb 1994.

[3] H. Benbrahim and J. A. Franklin, "Biped dynamic walking using reinforcement learning," *Robotics and Autonomous Systems*, vol. 22, pp. 283–302, 1997.

[4] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion." IEEE International Conference on Robotics and Automation, 2004.

[5] T. McGeer, "Passive dynamic walking," *International Journal of Robotics Research*, vol. 9, no. 2, pp. 62–82, April 1990.

[6] M. J. Coleman and A. Ruina, "An uncontrolled toy that can walk but cannot stand still," *Physical Review Letters*, vol. 80, no. 16, pp. 3658 – 3661, April 1998.

[7] H. Kimura and S. Kobayashi, "An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value functions." International Conference on Machine Learning (ICML '98), 1998, pp. 278–286.

[8] J. Baxter and P. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, vol. 15, pp. 319–350, 11 2001.

[9] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation." Advances in Neural Information Processing Systems, 1999.

[10] J. E. Wilson, "Walking toy," United States Patent Office, Tech. Rep., October 15 1936.

[11] R. Tedrake, T. W. Zhang, M. Fong, and H. S. Seung, "Actuating a simple 3d passive dynamic walker." IEEE International Conference on Robotics and Automation, 2004.

[12] R. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[13] S. H. Collins, M. Wisse, and A. Ruina, "A three-dimensional passive-dynamic walking robot with two legs and knees," *International Journal of Robotics Research*, vol. 20, no. 7, pp. 607–615, July 2001.