Learning to Walk in 20 Minutes

Russ Tedrake
Brain & Cognitive Sciences
Center for Bits and Atoms
Massachusetts Inst. of Technology
Cambridge, MA 02139
russt@csail.mit.edu

Teresa Weirui Zhang
Mechanical Engineering
Department
University of California, Berkeley
Berkeley, CA 94270
resa@berkeley.edu

H. Sebastian Seung
Howard Hughes Medical Institute
Brain & Cognitive Sciences
Massachusetts Inst. of Technology
Cambridge, MA 02139
seung@mit.edu

***Abstract*— We present a statistical gradient following algorithm which optimizes a control policy for bipedal walking online on a real robot. One of the distinguishing features of this system is that learning and execution occur simultaneously: there are no explicit learning trials and there is no need to model the dynamics of the robot in a simulation. Thanks in part to the mechanical design of the robot, the system is able to reliably acquire a robust policy for dynamic bipedal walking from a blank slate in less than 20 minutes. Once the robot begins walking, it quickly and continually adapts to the terrain with every step that it takes.**

I. Introduction

Despite many recent advances in robotics, there are still a number of fundamental motor tasks in which biological systems dramatically outperform robotic systems. Bipedal walking is one of those tasks. Some robotics researchers believe that an essential element missing in our robots is the ability to learn - to acquire and adapt motor behaviors through trial and error.

Over the last few years, we have begun to see examples of learning systems solving difficult control problems on real robotic systems. Atkeson and Schaal used learning from demonstration to train an anthropomorphic robotic arm to accomplish a pendulum swing-up task[1]. Morimoto and Doya used continuous-time temporal difference learning to teach a three-link robot to "stand up"[2]. Miller and Kun demonstrated the learning on a dynamic biped by coordinating hand-designed motor primitives[3]. More recently, there has been a great deal of work on learning fast gaits for Sony's AIBO (e.g. [4]). Ng and Thrun used an offline policy gradient algorithm to learn a policy for helicopter flight[5]. Although impressive, these results have not yet made a significant impact on the way that most robots are controlled.

II. The Walking Problem

A bipedal walking gait is considered dynamic if the ground projection of the center of mass leaves the convex hull of the ground contact points during some portion of the walking cycle. Achieving stable dynamic walking on a bipedal robot is a difficult control problem because bipeds can only control the trajectory of their center of mass through the unilateral, intermittent, uncertain force contacts with the ground. The walking control systems that dominate the robotics literature (e.g. [6]) restrict themselves to overly conservative gaits, and consequently cannot compare to human walking in terms of speed, efficiency, or robustness[7].

Dynamic walking is an excellent problem for studying learning control. Trial and error learning is a fundamentally different approach to achieving dynamic stability which could have an immediate and dramatic impact on the way that we control walking robots. Conversely, walking has all of the elements of a difficult learning problem. 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.

III. The Robot

In order to address the potential difficulties, we have carefully designed a bipedal robot which simplifies the learning problem. In particular, we have minimized the number of degrees of freedom on the robot down to a single joint in each hip and two joints in each ankle. More fundamentally, we have carefully designed the passive dynamics of the device to give it an important property: the ability to walk down a small ramp even when the control system is turned off. This design principle is known in the walking literature as *passive dynamic walking*, and it was originally inspired by children's walking toys ([8], [9]).

The walker shown on the left in Figure 1 is the simplest machine that we could build which is capable of stable passive dynamic walking in three dimensions. It has only a single passive pin joint at the hip. When placed at the

This work was supported by the National Science Foundation (grant CCR-022419), the David and Lucille Packard Foundation (contract 99-1471), and the Howard Hughes Medical Institute.

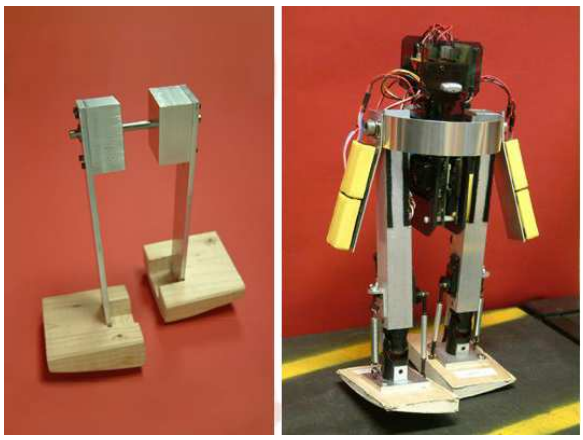

Fig. 1. The robot on the left is a simple passive dynamic walker. The robot on the right is our actuated version of the same robot.

top of a small ramp and given a small push, 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 [8]. 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 [10].

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 [10], 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 reduce the yaw moment created by the swinging leg.

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. Deriving a control policy for the ankle actuators turns out to be a very difficult problem because the robot is under-actuated - it has only four motors to control nine degrees of freedom[1]. This under-actuation means that the motors cannot produce an arbitrary acceleration of the robot at any instant in time and that the robot cannot follow an arbitrary trajectory, and therefore conventional robot manipulator control strategies do not apply. Before implementing any machine learning, we spent considerable time hand-designing control policies for this robot[10] to use as comparison for our learning experiments. Although we were able to stabilize the walking gait on flat terrain, the hand-designed controllers were not very robust nor efficient, and they required considerable tuning for each new surface that the robot walked on.

[1]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 does not slip, and we ignore the robot's absolute (x, y) position in space.

IV. The Learning Problem

In order to derive a control policy for this robot using reinforcement learning, we formulate the goal of control as stabilizing the limit cycle trajectory that the passive robot follows walking down the ramp, making it invariant to slope. We choose to characterize the stability of the limit cycle using a Poincaré map ([11], [12]) taken at the point when the robot's left leg collides with the ground. This allows us to replace the continuous-time dynamics of the limit cycle trajectories with a discrete footstep-to-footstep dynamics, and to formulate the learning on this discrete map.

More precisely, the dynamics of the robot can be written in the form

$$\ddot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}, \mathbf{d}(t)), \tag{1}$$

where $\mathbf{q}$ is the vector of joint angles denoting the state of the robot, $\mathbf{u}$ is the control vector, and $\mathbf{d}(t)$ is the time-varying vector of random disturbances. Our task is to acquire a deterministic feedback control policy

$$\mathbf{u} = \pi(\hat{\mathbf{q}}, \hat{\dot{\mathbf{q}}}), \tag{2}$$

where $\hat{\mathbf{q}}$ is a noisy estimate of $\mathbf{q}$ (due to noise in the sensors).

The Poincaré map, or return map, dynamics are a Markov random sequence with the probability at the $(n+1)$th footstep given by

$$\mathbf{F}_\pi(\mathbf{x}', \mathbf{x}) = P\left\{\hat{\mathbf{X}}_{n+1} = \mathbf{x}' | \hat{\mathbf{X}}_n = \mathbf{x}, \pi\right\}, \tag{3}$$

where $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^T$ evaluated at the return map. For clarity, we will reserve the variable $\mathbf{x}$ for describing the state of the robot on the return map. $\mathbf{F}_\pi(\mathbf{x}', \mathbf{x})$ represents the probability density function over the state space which contains the closed-loop dynamics of the robot integrated from one footstep to the next. We do not make any assumptions about its form, except that it is Markov. The stochasticity in $\mathbf{F}_\pi$ comes from the random disturbances, $\mathbf{d}(t)$, and from the sensor noise, $\hat{\mathbf{x}} - \mathbf{x}$. We describe the dynamics as a Markov sequence that is parameterized by the policy π instead of a Markov decision process (MDP) because the policy is evaluated many times between each crossing of the return map.

In this formulation, we reward the global stability of the desired limit cycle trajectory by penalizing the Euclidean

distance of each crossing of the return map from the desired crossing. This corresponds to an optimal control problem with the instantaneous cost using a constant desired value, $\mathbf{x}^d$, on the return map:

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

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 return map trajectory $\hat{\mathrm{x}} = [\hat{\mathbf{x}}(0), \hat{\mathbf{x}}(1), ..., \hat{\mathbf{x}}(N)]$, we define the average cost

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

Our goal is to find a policy π which minimizes the infinite-horizon average cost

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

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

V. The Learning Algorithm

To acquire a good control policy, we parameterize π using a function approximator that is linear in the parameters. The function approximator is parameterized by the vector $\mathbf{w}$ and uses nonlinear features ϕ:

$$\mathbf{u} = \pi_{\mathbf{w}}(\hat{\mathbf{x}}) = \sum_i w_i \phi_i(\hat{\mathbf{x}}). \quad (7)$$

The learning algorithm is a statistical actor-critic algorithm[13] 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 an online learning rule which changes $\mathbf{w}$ with each footstep that the robot takes. The form of the algorithm that we present here was originally proposed by [14]. We present a thorough derivation of the algorithm in [15].

The hallmark of the actor-critic family of reinforcement learning algorithms is the use of an approximation of the value function, $J^\pi(\mathbf{x})$, to improve the estimates of the performance gradient, and thereby improve the convergence rate of the algorithm. The value function is defined as the expected average cost to be incurred by following policy $\pi_{\mathbf{w}}$ starting from state $\mathbf{x}$:

$$J^{\pi_{\mathbf{w}}}(\mathbf{x}) = \lim_{N\to\infty} \frac{1}{N}\sum_{n=0}^{N} E\{g(\mathbf{x}(n))\}, \text{ with } \mathbf{x}(0) = \mathbf{x}. \quad (8)$$

We will use the shorthand $J^{\mathbf{w}}(\mathbf{x})$ for $J^{\pi_{\mathbf{w}}}(\mathbf{x})$. We use another function approximator, parameterized by the vector $\mathbf{v}$, to estimate the true value function:

$$\hat{J}^{\mathbf{w}}_{\mathbf{v}}(\hat{\mathbf{x}}) = \sum_i v_i \psi_i(\hat{\mathbf{x}}). \quad (9)$$

During learning, we estimate the performance gradient by varying $\mathbf{w}$, adding stochasticity to our deterministic policy. 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, $\mathbf{e}(n)$, called 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:

$$\delta(n) = g\left(\hat{\mathbf{x}}(n)\right) + \gamma\hat{J}_{\mathbf{v}}\left(\hat{\mathbf{x}}(n+1)\right) - \hat{J}_{\mathbf{v}}\left(\hat{\mathbf{x}}(n)\right) \quad (10)$$

$$e_i(n) = \gamma e_i(n-1) + b_i(n)z_i(n) \quad (11)$$

$$\Delta w_i(n) = -\eta_w\delta(n)e_i(n) \quad (12)$$

$$\Delta v_i(n) = \eta_v\delta(n)\psi_i(\hat{\mathbf{x}}(n)). \quad (13)$$

$\eta_w \geq 0$ and $\eta_v \geq 0$ are the learning rates and γ is the discount factor of the eligibility trace, which parameterizes the bias-variance trade-off of our gradient estimate[16]. $b_i(n)$ 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 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 $\hat{J}^{\mathbf{w}}_{\mathbf{v}}(\mathbf{x})$. 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. Simulataneously, the value estimate is updated using the temporal-difference learning algorithm, TD(0) [17]. This simple online algorithm performs approximate stochastic gradient descent on the expected value of the average infinite-horizon cost.

VI. Learning Implementation

The major limitation of the algorithm presented here is convergence rate. When $\mathbf{x}$ is high dimensional, then it requires many trials to effectively sample the entire state space. In order to emphasize fast convergence, we added a manual dimensionality reduction step which decomposes the control problem in the frontal and sagittal planes and exploits the dynamic coupling between the joint variables on the robot.

In the planar 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 in the sagittal plane, which effectively 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.

In the frontal plane, we constrain the feedback policy to be a function of only two variables: the roll angle of the body (θ_{roll}) and its derivative ($\dot{\theta}_{roll}$). The choice of these two variables is not arbitrary; they are the only variables that we used when writing the non-learning feedback controllers that stabilize 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. Because the value function is evaluated on the return map, it has one less dimension than the policy, making it a function of only a single variable: $\dot{\theta}_{roll}$. This dimensionality reduction step allows the algorithm to train very quickly, by manually imposing a mechanism of generalization over the state space and reducing the number of trials necessary to explore the space.

The control policy and value functions are both represented using linear function approximators of the form described in Equations 7 and 9, 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 $\hat{J}^{\mathbf{w}}_{\mathbf{v}}(\mathbf{x}) \approx \frac{g(\mathbf{x})}{1-\gamma}$. 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.

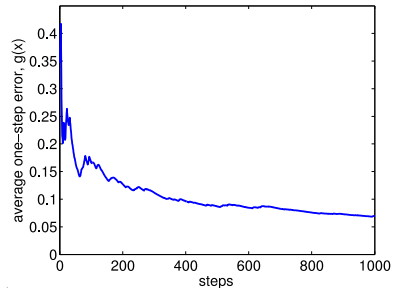

Fig. 2. A typical learning curve, plotted as the average error on each step.

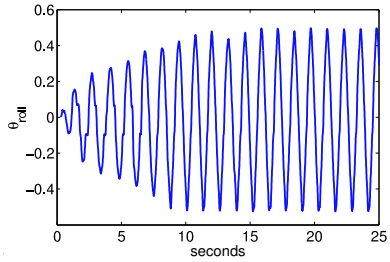

Fig. 3. θ_{roll} trajectory of the robot starting from standing still.

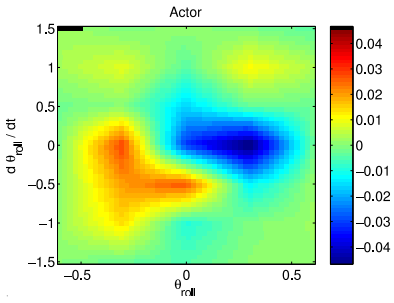

Fig. 4. Learned feedback control policy $u_{raRoll} = \pi_{\mathbf{w}}(\hat{\mathbf{x}})$.

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.

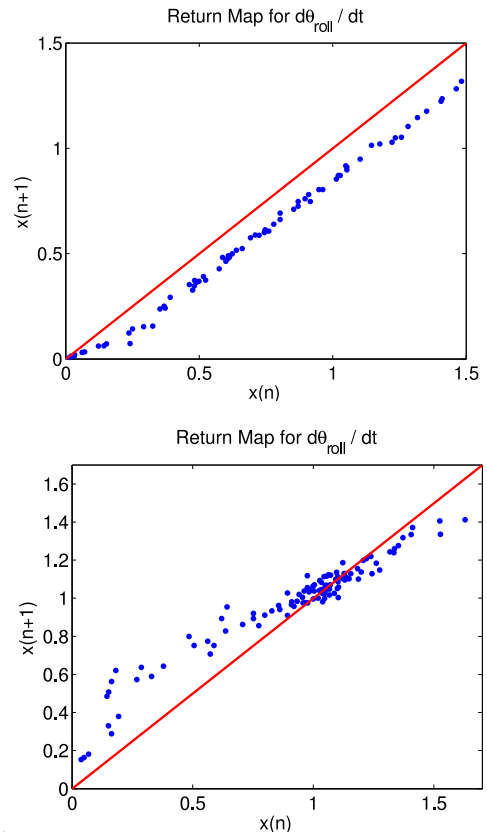

Fig. 5. Experimental return maps, before (top) and after (bottom) learning. Fixed points exist at the intersections of the return map (blue) and the line of slope one (red).

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 [10]) 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 by accumulating the eligibility within a step and discounting eligibility between steps. Interestingly, 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 [18], 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.

One of the distinguishing features of this learning system is that learning and execution occur simulatenously. The learning system is always on, and the robot makes small changes to its parameters with every step that it takes. The changes to the parameters on every step are small enough that they do not break the walking gait, in fact they are rarely even perceptible to the human eye, but they are large enough to measure a change in performance using the sensors. When the robot walks off a tiled surface onto a carpetted surface, the policy parameters change quickly and the robot is able to recover its gait within a dozen steps. This is an encouraging feature because it suggests that we use a learning system to augment an existing controller to allow that controller to systematically improve with experience.

Movies of Toddler can be found at:
http://hebb.mit.edu/people/russt/

IX. Acknowledgements

Special thanks to Ming-fai Fong, Derrick Tan, and Andrew Baines for their help with designing and building the Toddler robot.

References

[1] CG Atkeson and Stefan Schaal. Learning tasks from a single demonstration. *IEEE International Conference on Robotics and Automation (ICRA)*, 2:1706–1712, 1997.

[2] Jun Morimoto and Kenji Doya. Reinforcement learning of dynamic motor sequence: Learning to stand up. pages 1721–1726. IEEE/RSJ International, 1998.

[3] Andrew L. Kun and W. Thomas Miller, III. Adaptive dynamic balance of a biped robot using neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 240–245, 1996.

[4] Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[5] Andrew Y. Ng, H. Jin Kim, Michael I. Jordan, and Shankar Sastry. Autonomous helicopter flight via reinforcement learning. *Advances in Neural Information Processing Systems (NIPS)*, 16, 2003.

[6] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of honda humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1321–1326, 1998.

[7] Steven H. Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307:1082–1085, February 18 2005.

[8] John E. Wilson. Walking toy. Technical report, United States Patent Office, October 15 1936.

[9] Tad McGeer. Passive dynamic walking. *International Journal of Robotics Research*, 9(2):62–82, April 1990.

[10] Russ Tedrake, Teresa Weirui Zhang, Ming-fai Fong, and H. Sebastian Seung. Actuating a simple 3D passive dynamic walker. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 4656–4661, New Orleans, LA, April 2004.

[11] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Perseus Books, 1994.

[12] Daniel E. Koditschek and Martin Buehler. Analysis of a simplified hopping robot. *International Journal of Robotics Research*, 10(6):587–605, Dec 1991.

[13] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. *Advances in Neural Information Processing Systems*, 12:1008–1014, 1999.

[14] Hajime Kimura and Shigenobu Kobayashi. An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 278–286, 1998.

[15] Russ Tedrake. Probabilistic stability in legged systems and the mean first passage time (fpt) stability margin. *To be submitted to IEEE Transactions on Robotics*, 2004.

[16] J. Baxter, P.L. Bartlett, and L. Weaver. Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research 15 (2001) 351-381 Submitted 9/00; published 11/01*.

[17] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[18] Steven H. Collins, Martijn Wisse, and Andy Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *International Journal of Robotics Research*, 20(7):607–615, July 2001.