Reinforcement Learning Methods to Enable Automatic Tuning of Legged Robots

Nicolas Zeitlin
Pieter Abbeel, Ed.
Ronald S. Fearing, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-129
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-129.html

May 30, 2012

ABSTRACT

Reinforcement Learning Methods to Enable Automatic Tuning of Legged Robots

by

Nicolas Zeitlin

Masters of Engineering in EECS

University of California, Berkeley

Professor Pieter Abbeel

Reinforcement learning applied to legged-robots opens up the possibility to design robots capable not simply of walking, but of adapting and learning how to walk autonomously without any human interaction. This new generation of robots can one day navigate disaster areas and explore unchartered terrain. In this paper we evaluate the need for a reinforcement learning algorithm to optimize the gait of OctoRoACH, a hand-sized eight-legged robot. We then perform an evaluation of a likelihood-ratio gradient policy and compare it against our hand-tuned results. Finally, we suggest a different approach to reduce the policy search space that can make the problem more manageable.

I. Introduction

SEARCH and rescue operations present a challenging situation for rescuers, who must find and rescue the highest number of survivors while avoiding risking their own lives. In these situations time is of the essence, since survival rates plummet over time as survivors struggle trapped in the debris [1]. In order to reduce both the time necessary to find survivors and any risk for rescue personnel, robots are being developed to assist in the search and rescue operations. These remotely operated robots are tasked with navigating through rubble in search for survivors, allowing rescuers to monitor the situation safely from a distance.

Currently available rescue robots are remotely operated which reduces the amount that can be deployed and, in consequence, the speed at which survivors can be found. In this paper we present a technology that could allow a rescue robot to efficiently walk autonomously across the challenging disaster zone terrains, reducing its dependence from remote operators.

Legged robots, such as the DynaRoACH [5] or OctoRoACH [6] developed by the Biomimetic Millisystems lab, have demonstrated the ability to walk over a variety of terrains such as those encountered in a disaster situation. However, the efficiency and speed of their locomotion varies greatly according to the timing of their legs. Because this timing is also affected by several external factors, it is time-consuming, labor-intensive, and error-prone to tune by hand. Thus we first want to determine whether it is possible to hand-tune a single policy that can enable the robot to navigate across several terrains.

Our main contributions are to establish the need for a reinforcement learning algorithm to control the OctoRoACH's gait, and then specifically evaluate the performance of a Likelihood-Ratio Policy Gradient enabled controller. In particular, we first define how to measure OcotoRoACH's

walking efficacy. Then we manually optimize the robot's walking parameters around that definition and compare the results with those found by iteratively searching for an optimal policy through the Likelihood-Ratio Policy Gradient method.

The remainder of this paper is structured as follows. Section II describes previous work in the reinforcement learning space applied to legged robots. Section III describes our tests and policy gradient controller chosen parameters. Section IV exposes our results, and Section V discusses suggested future work.

II. PREVIOUS WORK

Given a legged robot's potential to navigate a wide variety of terrains, many efforts have been devoted lately to their development. For example Sony's Aibo[1] was a four-legged robot (popularized by the RoboCup) that was able to walk through different playing fields with varying levels of hardness and friction. LittleDog[2] in turn is a bio-inspired robot developed by Boston Dynamics, Inc. after a dog that was able to successfully traverse highly irregular terrain [2] [3]. Along the lines of bio-inspired robots, DASH [4] was inspired by cockroaches and offers a sprawled posture and uses an alternating tripod gait to achieve dynamic open-loop horizontal locomotion. After DASH other cockroach like robots have been developed including DynaRoACH and OctoRoACH, the latter of which has eight legs driven by two independent motors, each of which actuates all legs on one side. Having these two independent input controllers, as well as embedded PID controllers, makes OctoRoACH more powerful than its insect-inspired predecessors while remaining durable and inexpensive. For these reasons, we chose OctoRoACH to test the reinforcement learning algorithm evaluated in this paper. An

[1] http://www.sony.co.uk/support/en/hub/ERS
[2] http://www.bostondynamics.com/robot_littledog.html

additional feature for OctoRoACH considered recently also studies the possibility of adding a tail to the robot [6]. This method proved particularly successful when very quick turning was needed. Since we aren't particularly interested in fast turning of the robot, the simpler leg velocity control method was deemed sufficient for the work done here.

Given their potential to traverse different surfaces, enabling such legged-robots with the ability to walk efficiently has received a lot of attention in recent years by the robotics community. In [2] a hierarchical control architecture is presented that enables LittleDog to walk over rough terrain. This controller consists of a high-level planner that plans a set of footsteps across the terrain, a low-level planner that plans trajectories for the robot's feet and center of gravity (COG), and a low-level controller that tracks these desired trajectories using a set of closed-loop mechanisms. The high-level controller uses Hierarchical Apprenticeship Learning (HAL) to calculate the intended position of each foot throughout the path. This method was tested using LittleDog on four different tracks, two of which were built by the LittleDog program. The resulting success probabilities reached 97.5% with 20 runs.

The controller's architecture can clearly provide improvements, but it is also subject to the quality of learning algorithm. A popular approach to this problem is to compute the policy gradient that maximizes a reward based on the walking efficiency.

Policy Gradient methods in general have been widely popular in robotics as evidenced by [7]-[13]. These methods in turn can be usually grouped based on how they compute the policy gradient: by finite-difference or by Likelihood Ratio / REINFORCE methods [14].

Finite-difference methods start by varying a policy parameter by small increments of $\Delta\theta_i$ around a reference policy θ_h and evaluate the reward changes of $\theta_h + \Delta\theta_i$. These results are then used to

calculate the policy gradient of the chosen parameter using $\Delta J = J(\theta_h + \Delta\theta_i) - J(\theta_h)$. There have also been improved methods developed around variations of this approach. For instance in [15] a method is presented where random perturbations are applied simultaneously to the different dimensions of the policy, and the results grouped based on how much they improve or worsen the reward. This method can be used to arrive at a local optimum with less runs than with the traditional finite-difference approach. It was applied to optimize the speed of Aibo and yielded the best results up to that moment. PEGASUS [16], another popular approach, has also been described as building on a finite-difference gradient method [14]. This method in turn was applied to a bicycle simulator where the goal is to ride to a goal 1 Km. away. Using PEGASUS yielded results with a median riding distance ranging from 0.995 Km. to 1.07 Km, far better than the previous results published in [17].

Likelihood Ratio methods in general and REINFORCE [17] in particular arise from a different premise: given a set of trajectories τ generated stochastically with $\tau \sim p_\theta(\tau) = p(\tau|\theta)$ and a set of rewards $r(\tau) = \sum_{k=0}^{H} a_k r_k$ where H denotes the horizon, r_k represents the reward received by the learning system at time k and a_k denote time-step dependent weighting factors, the policy can be estimated by:

$$\nabla_\theta J(\theta) = \int_{\mathrm{T}} \nabla_\theta p_\theta(\tau) r(\tau) d\tau = E\{\nabla_\theta \log p_\theta(\tau)\, r(\tau)\} \text{ [9]}$$

This equation can be further optimized to reduce the variance by adding a baseline: since $\int_{\mathrm{T}} \nabla_\theta p_\theta(\tau) d\tau = 0$ we can add $b\nabla_\theta \log p_\theta(\tau)$ where b is a vector computed to minimize variance without biasing our estimates [10]. Likelihood ratio methods are also guaranteed to converge to the true gradient at the best Monte-Carlo convergence rate of $O\left(I^{-\frac{1}{2}}\right)$, where I denotes the number of roll-outs [19].

In [20] a further improvement is described for Likelihood Ratio policy methods. By making an analogy with importance sampling an optimization is described that leverages past experiences when computing the current gradient rewards. This optimization could enable a faster convergence rate by leveraging more data points therefore better estimating the current policy gradients.

Given the advantages of using policy gradient methods in general, and likelihood ratio methods with the importance sampling optimization in particular, these methods are good candidates to optimize legged-robots walking performance.

III. METHODOLOGY

To understand how our reinforcement learning algorithm improves the walking efficiency of OctoRoACH we chose to evaluate how its speed and accuracy are affected by our program. Currently, the available controller libraries offer functions to set the thrust for each of the two motors for a fixed amount of time. As mentioned above, each motor controls the left and right legs respectively. We chose to set the thrusts for a fixed amount of time and measure the distance traveled by the robot to assess its speed and its normalized deviation from a straight line to assess its accuracy. Specifically:

$$d(x, y) = \sqrt{x^2 + y^2}$$

$$s(d, t) = \frac{d}{t}$$

$$a(y, d) = \frac{y}{d}$$

In the equations above x and y are the robot's final coordinates taken with respect to its original position, d is the distance travelled, s is the measured speed during a fixed time t and a is the robot's path accuracy. Our intention is to maximize the accuracy and speed that the robot can achieve.

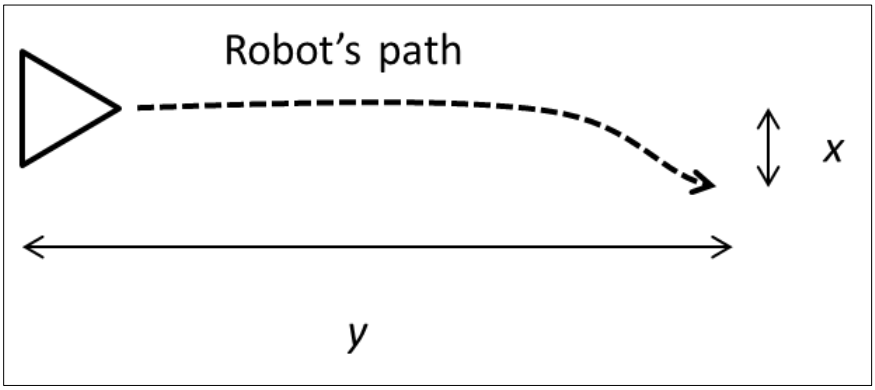

Figure 1: Measurements taken after each successful run. y corresponds to the straight distance travelled, while x corresponds to the perpendicular deviation

To understand OctoRoACH's range of policies and their effects on speed and accuracy, we tested a broad range of different hand-tuned thrust values on three different surfaces. In particular, we ran our tests on carpet, wood, and gravel. For each one of these surfaces we tried eight different thrust values, and to accommodate for random behavior and noise we tried perturbations of each thrust value six times. These runs were used not just to determine the relationship between speed and accuracy, but also as control samples to evaluate our algorithm against. To be considered a valid alternative, the reinforcement learning algorithm's policies should perform equal or better than the hand-tuned policies given a minimum accuracy rate.

We then proceeded to code the Likelihood-Ratio Policy Gradient Controller as described in [20]. Our reward function was determined by the harmonic mean of speed and accuracy, so as to

encourage policies having high values for both measurements. Specifically, our reward function was given by:

$$j = \frac{2 \times s(d,t) \times a(y,d)}{s(d,t) + a(y,d)}$$

$$j = \frac{2 \times \frac{d}{t} \times \frac{y}{d}}{\frac{d}{t} + \frac{y}{d}} = \frac{2yd}{d^2 + yt}$$

To minimize the impact of noise, we ran each policy six times. For each run, slightly perturbed input values were chosen using a multivariate distribution around the policy's thrust. To reduce the variance we decided to also use baselines as described in [20]. The baselines were calculated as the average reward for each given policy. Finally, the least squares method was used to calculate the policy gradient from the six distinct measurements. In other words, given *L* and *R* as the thrust on both motors, *j* as the reward function and $\vec{b}$ as the baseline, the least squares method was used to solve the following equation:

$$\begin{bmatrix} L_1 & R_1 \\ L_2 & R_2 \\ L_3 & R_3 \\ L_4 & R_4 \\ L_5 & R_5 \\ L_6 & R_6 \end{bmatrix} \nabla f = \begin{bmatrix} j_1 \\ j_2 \\ j_3 \\ j_4 \\ j_5 \\ j_6 \end{bmatrix} - \vec{b}$$

The gradient was then used to calculate new thrust values by following the direction of greatest ascent. We then proceeded to repeat this process running OctoRoACH on one of the chosen surfaces, carpet, until the control policies converged and compared the resulting speed and accuracy against the hand-tuned policies. To determine that a policy had converged we waited until the integer thrust values chosen by the algorithm were the same three times in a row.

IV. Results and Analysis

The hand-tuned experiments were useful to confirm the need for a reinforcement learning algorithm for policy control. We found that there was no single policy with a high speed and accuracy that could run equally well on all different surfaces. In fact, those policies that had returned a high reward value on a given surface often had a very poor performance on all others. The plot below illustrates this by contrasting the speed and accuracy of a set of policies on wood and carpet.

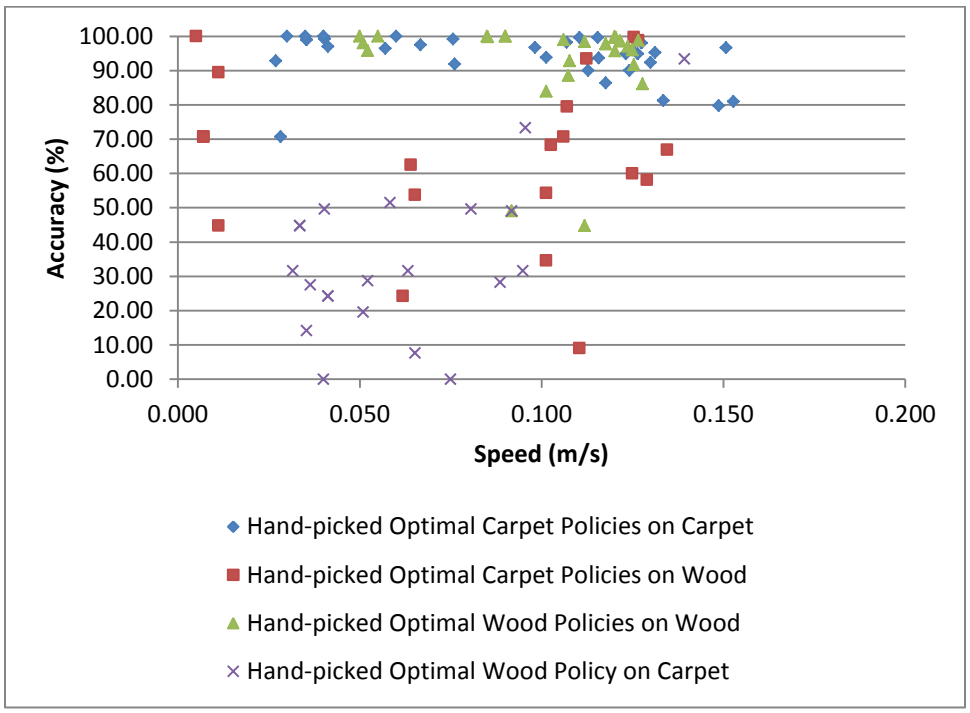

Figure 2: Speed and accuracy for 2 different sets of policies, evaluated on carpet and wood. Note how the same policies that yield high rewards on one surface may have a poor performance on another

The experiments were also useful to understand how speed and accuracy relate to each other: as speed increases, we found that the path's accuracy may decrease, as can be seen in Figure 3 below.

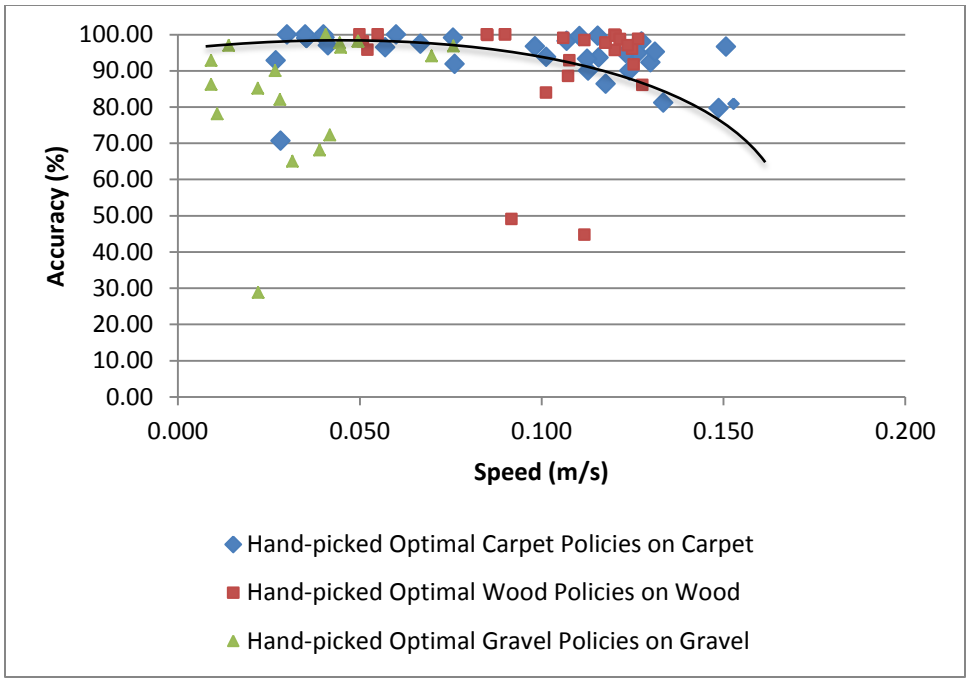

Figure 3: Hand-tuned Optimized Speed and Accuracy measurements for different policies and different surfaces.

This tradeoff between speed and accuracy confirmed our assumptions and our reasoning for choosing the harmonic mean between the normalized speed and accuracy as the reward function.

As a result, when we executed the likelihood-ratio policy gradient experiments on a carpet surface, we were able to reach speeds and accuracies similar to those found manually and shown in the plots above. After 8 iterations (with six runs each) the speed and accuracy converged to 0.1*m/s* and 99.5% respectively. A plot of the reward function shows the convergence, in spite of choosing a poorly performing policy in the fourth iteration.

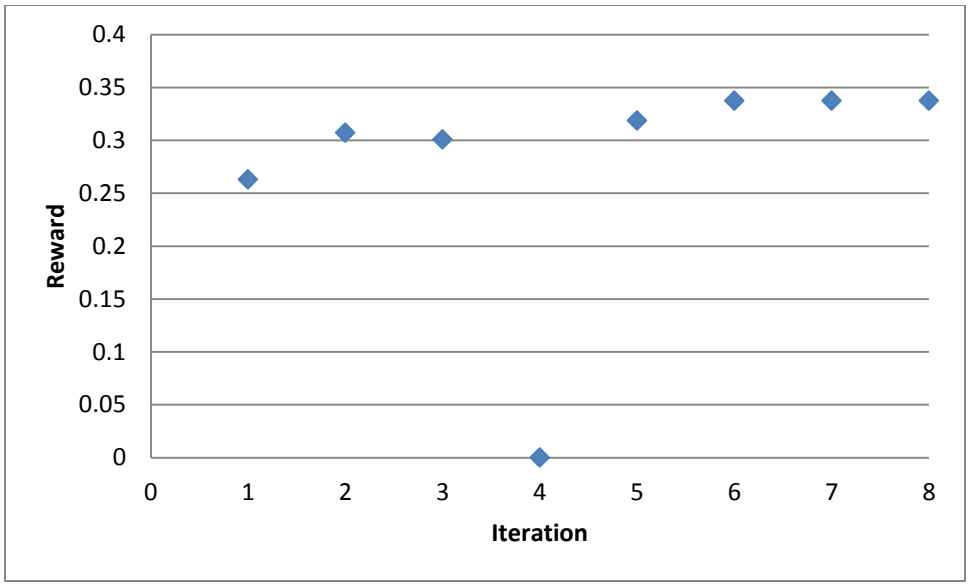

Figure 4: Rewards computed by the Policy Gradient Algorithm. In spite of running into a poorly performing policy in the fourth iteration, the algorithm later converged

V. LIMITATIONS AND FUTURE WORK

More tests should be performed to confirm the effectiveness of the reinforcement learning algorithm. In particular, it would be useful to assess the validity of the algorithm on additional surfaces. For multiple surfaces, one possibility would be to establish a mapping between optimal policies across those surfaces. If equivalent rewards across surfaces can be found merely modulating and translating through the 2-dimensional policy space, then a policy gradient algorithm could be customized to find the appropriate translation.

To establish whether such a mapping is possible, we measured the robot's distance travelled across a longitudinal axis over a fixed amount of time using different policies on carpet and wood. The resulting plots are shown below.

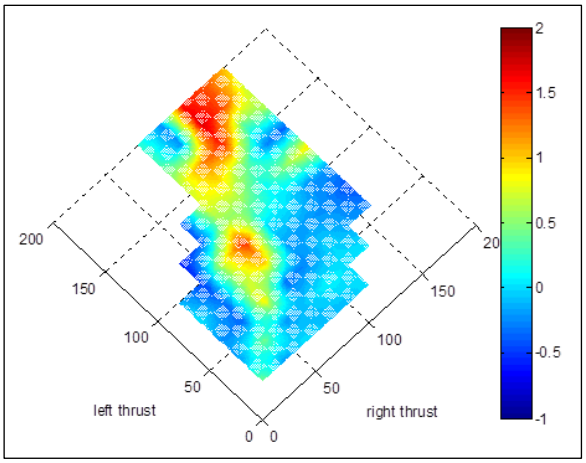

Figure 5: Distance travelled over a wooden given a range of thrust values

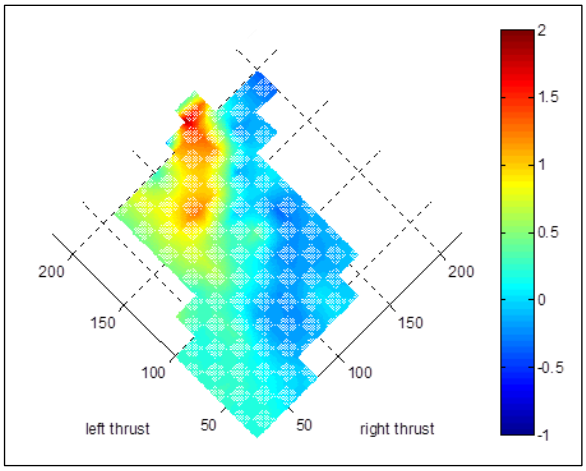

Figure 6: Distance travelled over carpet surface given a range of thrust values

As can be inferred from the plots above, there is a relationship between the resulting distances travelled on wood (Figure 5) and carpet (Figure 6). Specifically, we see that similar distances can be reached on carpet as in wood by increasing the left and right thrust values by 75 and 50 points respectively. If this pattern were consistent across additional surfaces, then the policy search could be performed in a lower dimensional space.

VI. Conclusions

By evaluating different walking policies on different surfaces, we were able to establish the necessity for a reinforcement learning enabled controller if we want OctoRoACH to efficiently traverse different terrains. One possible method is Likelihood-Ratio Policy Gradient algorithm, which we have shown has the ability to optimize the robots' gait and reach speeds and accuracies similar to those found by manual hand-tuning. This suggests that a walking controller for OctoRoACH would benefit from such an algorithm, and encourages further research in mapping the policy space to develop better policy gradients.

REFERENCES

[1] A.W. Coburn, R.J.S. Spence, A. Pomonis, "Factors determining human casualty levels in earthquakes: Mortality prediction in building collapse" in *Earthquake Engineering, Tenth World Conference, Balkema, Rotterdam*, 1992.

[2] J. Zico Kolter, Mike P. Rodgers, and Andrew Y. Ng. "A control architecture for quadruped locomotion over rough terrain". In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 811-818, 2008

[3] J. R. Rebula, P. D. Neuhaus, B. V. Bonnlander, M. J. Johnson, and J. E. Pratt, "A controller for the littledog quadruped walking on rough terrain". In *Proceedings of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2007

[4] P. Birkmeyer, K. Peterson, and R. S. Fearing, "Dash: A dynamic 16g hexapedal robot", in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2009

[5] A. Hoover, S. Burden, X.-Y. Fu, S. Sastry and R. Fearing, "Bio-inspired design and dynamic maneuverability of a minimally actuated six-legged robot" in *IEEE International Conference on Biomedical Robotics and Biomechatronics*, BioRob 2010., Sep. 2010

[6] A.O. Pullin, N.J. Kohut, D. Zarrouk and R. Fearing, "Dynamic turning of 13 cm robot comparing tail and differential drive"

[7] T. Mori, Y. Nakamura, M. Sato, and S. Ishii. "Reinforcement learning for cpg-driven biped robot". In *AAAI*, 2004

[8] R. Tedrake, T. W. Zhang, and H.S. Seung. "Learning to walk in 20 minutes". In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, 2005

[9] J. Baxter and P. Bartlett. "Direct gradient-based reinforcement learning". In *Journal of Artificial Intelligence Research*, 1999

[10] E. Greensmith, P. Bartlett, and J. Baxter. "Variance reduction techniques for gradient estimates in reinforcement learning". In *Journal of Machine Learning Research*, 2004

[11] Leonid Peshkin and Christian R. Shelton. "Learning from scarce experience". In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002

[12] R. Sutton, D. McAllester, S. Singh, and Y. Mansour. "Policy gradient methods for reinforcement learning". In *NIPS 13*, 2000

[13] Jens Kober and Jan Peters. "Policy search for motor primitives in robotics". In *NIPS*, 2008

[14] J. Peters and S. Schaal. "Policy gradient methods for robotics". In *Proceedings of the IEEE International Conference on Intelligent Robotics Systems*, 2006

[15] N. Kohl and P. Stone, “Policy gradient reinforcement learning for fast quadrupedal locomotion”. In *IEEE International Conference on Robotics and Automation*, 2004

[16] Andrew Y. Ng and Michael Jordan. “PEGASUS: A policy search method for large MDPs and POMDPs“. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 406–415, Stanford, California, 2000

[17] J. Randløv and P. Alstrøm. “Learning to drive a bicycle using reinforcement learning and shaping”. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998

[18] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” In *Machine Learning*, vol. 8, no. 23, 1992

[19] P. Glynn, “Likelihood ratio gradient estimation: an overview”. In *Proceedings of the 1987 Winter Simulation Conference*, Atlanta, GA, 1987, pp. 366–375

[20] J. Tang and P. Abbeel, “On a connection between importance sampling and the likelihood ratio policy gradient”. In *Neural Information Processing Systems*, 2011