Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion

Nate Kohl and Peter Stone
Department of Computer Sciences, The University of Texas at Austin
1 University Station C0500, Austin, Texas 78712-1188
{nate,pstone}@cs.utexas.edu
http://www.cs.utexas.edu/˜{nate,pstone}

Abstract— **This paper presents a machine learning approach to optimizing a quadrupedal trot gait for forward speed. Given a parameterized walk designed for a specific robot, we propose using a form of policy gradient reinforcement learning to automatically search the set of possible parameters with the goal of finding the fastest possible walk. We implement and test our approach on a commercially available quadrupedal robot platform, namely the Sony Aibo robot. After about three hours of learning, all on the physical robots and with no human intervention other than to change the batteries, the robots achieved a gait faster than any previously known gait for the Aibo, significantly outperforming a variety of existing hand-coded and learned solutions.**

Keywords: Learning Control, Walking Robots, Multi Legged Robots

I. Introduction

Locomotion of legged robots is a challenging multidimensional control problem. It requires the specification and coordination of motions in all of the robots' legs while accounting for factors such as stability and surface friction.

One popular research platform for legged locomotion currently is the Sony Aibo robot,[1] spurred on in part by the annual RoboCup 4-legged soccer competitions [1]. In this domain, the speed of individual robots is often a major factor in determining the success of a team. Since the default gait that comes with the Aibo is fairly slow, there has been significant incentive within the RoboCup community to develop improved locomotion for the Aibos.

Until recently, most of the locomotion improvements for the Aibo centered around hand-tuning a parameterized gait. This approach has been somewhat fruitful: since the inception of the RoboCup legged league in 1998, the speed of the Aibos has increased significantly. However, the process of hand-tuning a parameterized gait both can be time-consuming and can require a good deal of human expertise. Furthermore, a change of robot hardware and/or the surface on which it is to walk necessitates tuning anew.

One alternative to hand-tuning a parameterized gait is to use machine learning to automate the search for good parameters. In the past, various machine learning techniques have proven to be useful in finding control policies for a wide variety of robots including helicopters [2], [3], biped robots [4] and Aibos [5], [6], [7]. This paper presents a policy gradient reinforcement learning approach for generating a fast walk on legged robots. Using this method, we have created a walk that is faster than hand-tuned gaits and among the fastest learned gaits.

The remainder of this paper is organized as follows. Section II introduces the Sony Aibo ERS-210A robot platform and summarizes general methods for enabling Aibos to walk, both past and current. Section III presents our method for automatically generating a fast walk via machine learning, specifically policy gradient reinforcement learning. Section IV reports on our learning experiments. Section V presents discussion and future work and Section VI concludes.

II. Walking Aibos

The Sony Aibo ERS-210A is a commercially available robot that comes equipped with a color CMOS camera and an optional ethernet card that can be used for wireless communication. The Aibo is a quadruped robot, and has three degrees of freedom in each of its four legs.

At the lowest level, the Aibo's gait is determined by a series of joint positions for the three joints in each of its legs. An early attempt to develop a gait by Hornby et al.[2] involved using a genetic algorithm to learn a set of low-level parameters that described joint velocities and body position [5]. More recent attempts to develop gaits for the Aibo have involved adopting a higher-level representation that deals with the trajectories of the Aibo's four feet through three-dimensional space. An inverse kinematics calculation can then be used to convert these trajectories into joint angles.

Among higher-level approaches, most of the differences between gaits that have been developed for the Aibo stem from the shape of the loci through which the feet pass and the exact parameterizations of those loci. For example, a team from the University of New South Wales achieved the fastest known hand-tuned gait using the high-level approach described above with trapezoidal loci. They subsequently generated an even faster walk via learning [7]. In research conducted independently of the work presented here, the University of Newcastle team was able to generate very fast gaits by using a genetic algorithm and a similar loci parameterization [8]. A team from Germany created a flexible gait implementation that allows them to use a variety of different shapes of loci [9].

[1] http://www.aibo.com

[2] Developed on an earlier version of the Aibo.

Our team (UT Austin Villa [10]) first approached the gait optimization problem by hand-tuning a gait described by half-elliptical loci. This gait performed comparably to those of other teams participating in RoboCup 2003. The work reported in this paper uses the hand-tuned UT Austin Villa walk as a starting point for learning. Figure 1 compares the reported speeds of the gaits mentioned above, both hand-tuned and learned, including that of our starting point, the UT Austin Villa walk. The latter walk is described fully in a team technical report [10]. The remainder of this section describes the details of the UT Austin Villa walk that are important to understand for the purposes of this paper.

Hand-tuned gaits				Learned gaits	
CMU	German	UT Austin		Hornby	
(2002)	Team	Villa	UNSW	1999	UNSW
200	**230**	**245**	**254**	**170**	**270**

Fig. 1. Maximum forward velocities of the best current gaits (in mm/s) for different teams, both learned and hand-tuned.

The half-elliptical locus used by our team is shown in Figure 2. By instructing each foot to move through a locus of this shape, with each pair of diagonally opposite legs in phase with each other and perfectly out of phase with the other two (a gait known as a trot), we enable the Aibo to walk. Four parameters define this elliptical locus:

1) The length of the ellipse;
2) The height of the ellipse;
3) The position of the ellipse on the x axis; and
4) The position of the ellipse on the y axis.

Since the Aibo is roughly symmetric, the same parameters can be used to describe loci on both the left and right side of the body. To ensure a relatively straight gait, the length of the ellipse is the same for all four loci. Separate values for the elliptical height, x, and y positions are used for the front and back legs. An additional parameter which governs the turning rate of the Aibo is used to determine the skew of all four ellipses in the x-y plane, a technique introduced by the UNSW team [11].[3] The amount of skew is determined by the product of the angle at which the Aibo wishes to move and this skew multiplier parameter.

All told, the following set of 12 parameters define the Aibo's gait [10]:

- The front locus (3 parameters: height, x-pos., y-pos.)
- The rear locus (3 parameters)
- Locus length
- Locus skew multiplier in the x-y plane (for turning)
- The height of the front of the body
- The height of the rear of the body
- The time each foot takes to move through its locus
- The fraction of time each foot spends on the ground

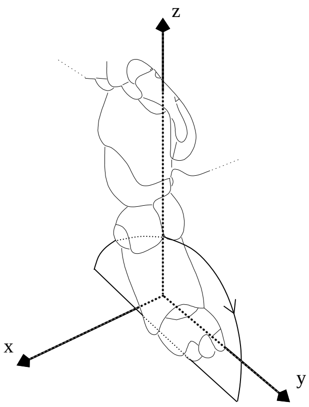

Fig. 2. The elliptical locus of the Aibo's foot. The half-ellipse is defined by length, height, and position in the x-y plane.

During the American Open tournament in May of 2003,[4] UT Austin Villa used a simplified version of the parameterization described above that did not allow the front and rear heights of the robot to differ. Hand-tuning these parameters generated a gait that allowed the Aibo to move at 140 mm/s. After allowing the front and rear height to differ, the Aibo was tuned to walk at 245 mm/s in the RoboCup 2003 competition.[5] Applying machine learning to this parameter optimization process, however, allowed us to significantly improve the speed of the Aibos, as described in the following section.

III. Learning the Walk

Given the parameterization of the walk defined in Section II, our task amounts to a parameter optimization problem in a continuous 12-dimensional space. For the purposes of this paper, we adopt forward speed as the sole objective function. That is, as long as the robot does not actually fall over, we do not optimize for any form of stability (for instance in the face of external forces from other robots).

We formulate the problem as a policy gradient reinforcement learning problem by considering each possible set of parameter assignments as defining open-loop policy that can be executed by the robot. Assuming that the policy is differentiable with respect to each of the parameters, we estimate the policy's gradient in parameter space, and then follow it towards a local optimum.

Since we do not know anything about the true functional form of the policy, we cannot calculate the gradient exactly. Furthermore, empirically estimating the gradient by sampling can be computationally expensive if done naively, given the large size of the search space and the temporal cost of each evaluation. Given the lack of accurate simulators for the Aibo, we are forced to perform the learning entirely on real robots, which makes efficiency a prime concern.

In this section we present an efficient method of estimating the policy gradient. It can be considered a degenerate

[3] Even when walking directly forward, noise in an Aibo's motions occasionally requires that the four ellipses be skewed to allow the Aibo to execute small turns in order to stay on course.

[4] `http://www.cs.cmu.edu/~AmericanOpen03/`

[5] `http://www.openr.org/robocup/`. Thanks to Daniel Stronger for hand-tuning the walks to achieve these speeds.

form of standard policy gradient reinforcement learning techniques [12], [13] in that the control policy is essentially open loop and that the only effect of sensory input is to make small steering adjustments to compensate for noise. Like these more general techniques, our approach will only converge towards a local optimum. In contrast, some action-value reinforcement learning algorithms, such as Q-learning provably converge to the globally optimal policy [14]. However, Q-learning, which is designed for Markov decision processes, is not directly applicable to our problem, which features open-loop control and no notion of "state".

Our approach starts from an initial parameter vector $\pi = \{\theta_1, \ldots, \theta_N\}$ (where $N = 12$ in our case) and proceeds to estimate the partial derivative of π's objective function with respect to each parameter. We do so by first evaluating t randomly generated policies $\{R_1, R_2, \ldots, R_t\}$ near π, such that each $R_i = \{\theta_1 + \Delta_1, \ldots, \theta_N + \Delta_N\}$ and each Δ_j is chosen randomly to be either $+\epsilon_j$, 0, or $-\epsilon_j$. Each ϵ_j is a fixed value that is small relative to θ_j. As described below, the evaluation of each policy generates a score that is a measure of the speed of the gait described by that policy.

After evaluating the speed of each R_i, we estimate the partial derivative in each of the N dimensions. We do this by grouping each R_i into one of three sets for each dimension n:

$$R_i \in \begin{cases} S_{+\epsilon,n} & \text{if the } n\text{th parameter of } R_i \text{ is } \theta_n + \epsilon_n \\ S_{+0,n} & \text{if the } n\text{th parameter of } R_i \text{ is } \theta_n + 0 \\ S_{-\epsilon,n} & \text{if the } n\text{th parameter of } R_i \text{ is } \theta_n - \epsilon_n \end{cases}$$

We then compute an average score $Avg_{+\epsilon,n}$, $Avg_{+0,n}$, and $Avg_{-\epsilon,n}$ for $S_{+\epsilon,n}$, $S_{+0,n}$, and $S_{-\epsilon,n}$, respectively. These three averages give us an estimate of the benefit of altering the nth parameter by $+\epsilon_n$, 0, or $-\epsilon_n$. Note that in expectation, there will be $t/3$ of the t policies with each of the three possible parameter settings, though there will be some random variation. For an example of this process for one dimension, see Figure 3. We use these scores to construct an adjustment vector A of size N, where

$$A_n = \begin{cases} 0 & \text{if } Avg_{+0,n} > Avg_{+\epsilon,n} \text{ and} \\ & \quad Avg_{+0,n} > Avg_{-\epsilon,n}, \\ Avg_{+\epsilon,n} - Avg_{-\epsilon,n} & \text{otherwise} \end{cases}$$

We normalize A and multiply it by a scalar step-size η, so that our adjustment will remain a fixed size each iteration. Finally, we add A to π, and begin the next iteration. Pseudocode for this policy gradient algorithm is shown in Figure 4.

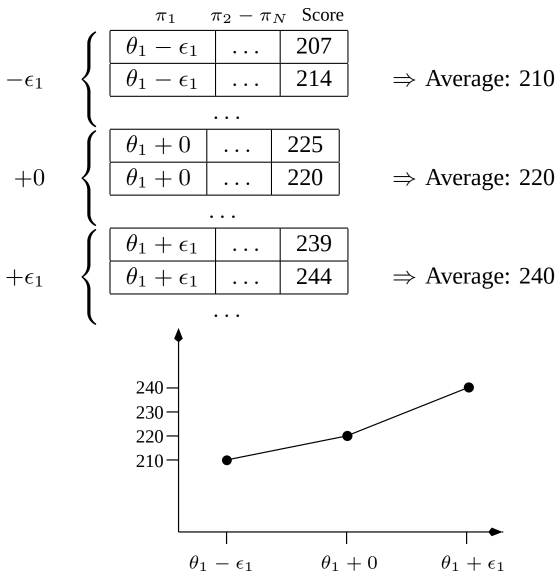

Fig. 3. An example of the process for estimating the gradient in one dimension. Each R_i is grouped into one of three categories, depending on the value of the first parameter (π_1) of each R_i. The averages for these categories can be used to estimate the value of the objective function at and around θ_1.

```
$\pi \leftarrow InitialPolicy$
while !done do
  $\{R_1, R_2, \ldots, R_t\} = t$ random perturbations of $\pi$
  evaluate( $\{R_1, R_2, \ldots, R_t\}$ )
  for $n = 1$ to $N$ do
    $Avg_{+\epsilon,n} \leftarrow$ average score for all $R_i$ that have a positive
                 perturbation in dimension $n$
    $Avg_{+0,n} \leftarrow$ average score for all $R_i$ that have a zero
                 perturbation in dimension $n$
    $Avg_{-\epsilon,n} \leftarrow$ average score for all $R_i$ that have a
                 negative perturbation in dimension $n$
    if $Avg_{+0,n} > Avg_{+\epsilon,n}$ and $Avg_{+0,n} > Avg_{-\epsilon,n}$ then
      $A_n \leftarrow 0$
    else
      $A_n \leftarrow Avg_{+\epsilon,n} - Avg_{-\epsilon,n}$
    end if
  end for
  $A \leftarrow \frac{A}{|A|} * \eta$
  $\pi \leftarrow \pi + A$
end while
```

Fig. 4. Pseudocode for the N-dimensional policy gradient algorithm. During each iteration of the main loop we sample t policies near π to estimate the gradient around π, then move π by an amount of η in the most favorable direction.

We controlled our experiments from a computer that was connected via wireless ethernet to the Aibos. All of the policy evaluations took place on actual robots, without the use of a simulator. Previous attempts at learning Aibo gaits involved running each experiment directly on the Aibo, which imposed certain time limitations on the learning process [7]. A more decentralized approach allowed us to distribute the learning process over multiple Aibos and prevented the loss of data due to battery swaps and mechanical failure. This also meant that the only human intervention required during an experiment involved replacing discharged batteries, an event which occurred about once an hour. We used three simultaneously walking Aibos for our experiments, but our approach is general and allows us to scale to arbitrary numbers of Aibos.

We evaluated the efficacy of a set of parameters by sending those parameters to an Aibo and instructing it to time itself as it walked between two fixed landmarks (Figure 5). More efficient parameters resulted in a faster gait, which translated into a lower time and a better score. After completing an evaluation, the Aibo sent the resulting score back to the host computer and prepared itself for a new set of parameters to evaluate.[6]

When implementing the algorithm described above, we chose to evaluate $t = 15$ policies per iteration. Since there was significant noise in each evaluation, each set of parameters was evaluated three times. The resulting score for that set of parameters was computed by taking the average of the three evaluations. Each iteration therefore consisted of 45 traversals between pairs of beacons and lasted roughly $7\frac{1}{2}$ minutes. Since even small adjustments to a set of parameters could lead to major differences in gait speed, we chose relatively small values for each ϵ_j. To offset the small values of each ϵ_j, we accelerated the learning process by using a larger value of $\eta = 2$ for the step size.

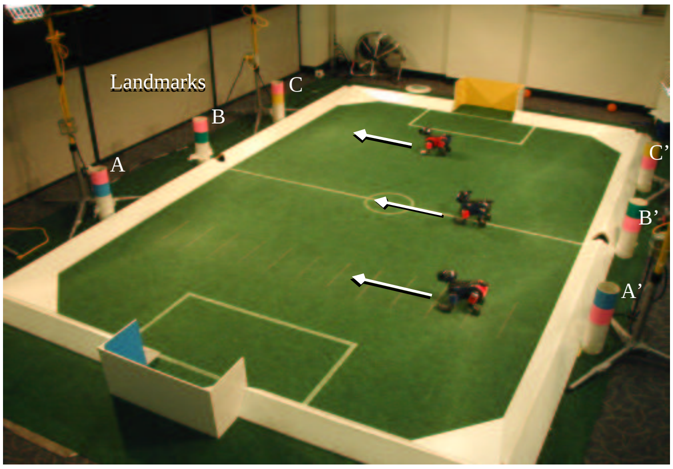

Fig. 5. The training environment for our experiments. Each Aibo times itself as it moves back and forth between a pair of landmarks (A and A', B and B', or C and C').

IV. Results

Our main result is that using the algorithm described in Section III, we were able to find one of the fastest known Aibo gaits. Figure 6 shows the performance of the best policy of each iteration during the learning process. After 23 iterations the learning algorithm produced a gait (shown in Figure 8) that yielded a velocity of 291 ± 3 mm/s, faster than both the best hand-tuned gaits and the best previously known learned gaits (see Figure 1). The parameters for this gait, along with the initial parameters and ϵ values are given in Figure 7.

Note that we stopped training after reaching a peak policy at 23 iterations, which amounted to just over 1000 field traversals in about 3 hours. Subsequent evaluations showed no further improvement, suggesting that the learning had plateaued.

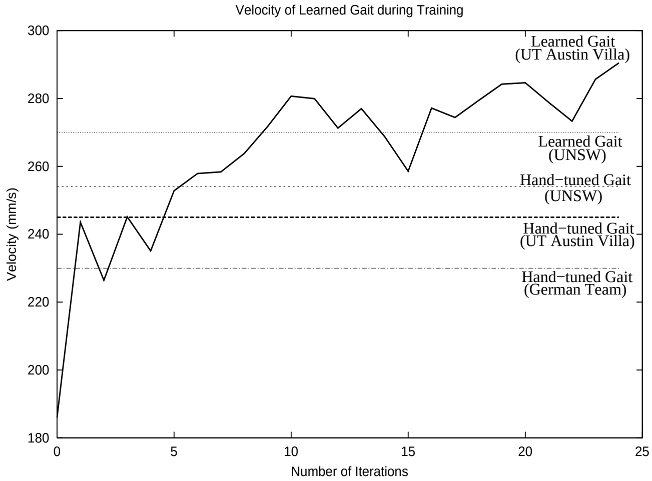

Fig. 6. The velocity of the best gait from each iteration during training, compared to previous results. We were able to learn a gait significantly faster than both hand-coded gaits and previously learned gaits.

Parameter	Initial Value	ϵ	Best Value
Front locus:			
(height)	4.2	0.35	4.081
(x offset)	2.8	0.35	0.574
(y offset)	4.9	0.35	5.152
Rear locus:			
(height)	5.6	0.35	6.02
(x offset)	0.0	0.35	0.217
(y offset)	-2.8	0.35	-2.982
Locus length	4.893	0.35	5.285
Locus skew multiplier	0.035	0.175	0.049
Front height	7.7	0.35	7.483
Rear height	11.2	0.35	10.843
Time to move through locus	0.704	0.016	0.679
Time on ground	0.5	0.05	0.430

Fig. 7. The initial policy, the amount of change (ϵ) for each parameter, and best policy learned after 23 iterations. All values are given in centimeters, except time to move through locus, which is measured in seconds, and time on ground, which is a fraction.

There are a couple of possible explanations for the amount of variation in the learning process, visible as the spikes in the learning curve shown in Figure 6. Despite the fact that we averaged over multiple evaluations to determine the score for each policy, there was still a fair amount of noise associated with the score for each policy. It is entirely possible that this noise led the search astray at times, causing temporary decreases in performance. Another explanation for the amount of variation in the learning process could be the relatively large step size ($\eta = 2$) used to adjust the policy at the end of each iteration. As mentioned previously, we chose a large step size to offset the relatively small values of ϵ. While serving to accelerate the rate of learning, this large step size might also have caused the search process to periodically jump into a

[6] There is video of the training process at: `www.cs.utexas.edu/~AustinVilla/legged/learned-walk/`

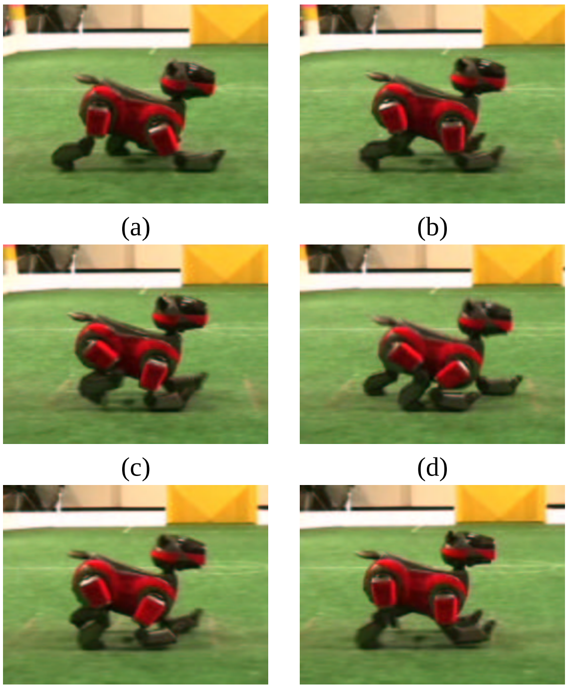
(a) (b) (c) (d) (e) (f)

Fig. 8. A series of snapshots of the best learned gait, which allows the Aibo to trot at 291 mm/s.

bad part of the policy space, which would have again caused a temporary decrease in performance.

Two possible reasons for the overall success of our experiment are (i) that our policy gradient algorithm is particularly effective for learning this task, and (ii) that the gait implementation itself is superior to previous approaches. To test the first possibility, we compared the performance of our approach to that of other parameter optimization methods, such as those presented by Press [15]. For example, Kim and Uther used Powell's method to tune their walk [7], and Weingarten et al. applied Nelder and Mead's *Downhill Simplex Method* (aka "amoeba") to generate the fastest known legged walk in terms of body lengths per second on a custom hexapod robot platform [16]. As a comparison point, we implemented the Nelder Mead algorithm and used it to generate gaits for our robots. Preliminary results with this method suggest that while it was able to do some learning, the volume of the simplex quickly converged to zero which required that the algorithm be restarted frequently. The fastest gait that this method was able to discover has a reasonable speed of 248 mm/s. So far, the policy gradient algorithm currently has yielded better results with fewer evaluations, suggesting that the particular learning algorithm we use does indeed play a role in our success.

It is also possible that our particular gait implementation was a key factor. Enabling an Aibo to walk requires careful crafting of a variety of components that must work together seamlessly. Since we developed all of our code from scratch, it is likely that some parts of our low-level gait implementation differed from other implementations. The aggregate effect of these differences could have some affect the process of learning an efficient gait. Unfortunately, it is difficult to determine the impact of any particular implementation, due to the difficulty of porting or exactly re-creating code from other implementations.

The starting point for the search could have also affected our final results. Since it was not obvious where the search algorithm should start, we tried several different starting points that varied in the degree to which they had been hand-tuned. We tested one extreme of this spectrum by starting from completely random parameters. This proved to be too much of an obstacle for the Aibos, since the gaits defined by random parameters often led to violent behaviors and frequent mechanical failures. When we started from parameters that had been slightly tuned by hand, we found that the learning algorithm was able to discover a gait that moved at 186 mm/s, but the rate of learning was too slow to be useful.

Conversely, when we started from parameters that had been finely hand-tuned, we found that the learning algorithm had to escape a local maximum before it could make any progress. Although this starting point yielded better results (260 mm/s) than both the random and the slightly hand-tuned starting points, our fastest result came from a compromise. The best starting point proved to be a set of parameters (shown in Figure 7) that had been roughly tuned by hand, but not overly tuned, and which constituted a reasonable but slow gait.

V. Discussion and Future Work

One of the useful aspects of automating the gait optimization process is that search algorithms often possess less bias than human engineers. For example, our gait was designed after a trot gait, where diagonally opposite legs strike the ground simultaneously. We assumed that the ideal trot gait would keep two feet on the ground at all times. Interestingly enough, the best learned gait defied our expectations by attempting to keep each foot on the ground only 43% of the time. By allowing the learning process to affect a large portion of the gait, we were able to discover parameter settings that we would not have likely found through hand-tuning.

On a similar note, recent work has suggested that due to mechanical limitations and environmental noise, the actual locus that each foot follows is significantly different than the requested half-elliptical locus [17]. Given this discrepancy between the ideal locus and the real locus, a change in parameters describing the ideal locus may not have the intended effect on the real locus. This discrepancy could make hand-tuning difficult for humans, who expect a certain correspondence between parameters and their effects on the gait. Since the learning process is unaware of the semantics of the parameters, it therefore might not suffer as much from discrepancies between the expected loci and the actual loci.

Another benefit of automated learning can arise in situations such that the robots are required to repeatedly switch surfaces. In RoboCup, the surfaces of different playing fields can vary widely in hardness and friction. Repeatedly tuning parameters by hand for each surface could consume a great deal of time from human engineers, whereas automatically learning parameters for various surfaces would require significantly less human intervention.

Our method could be considered a form of multi-robot learning, in that it would be relatively straightforward to implement the learning algorithm presented here without the use of a central controller. Since the evaluations that each Aibo performs are generated randomly, the algorithm that each Aibo executes could be run almost completely independently of the other Aibos. The only requirement would be that each Aibo communicate the results of the evaluations that it performs to the other Aibos and that they have a common notion of the parameter t which governs the number of policies evaluated per iteration. After finding out the results of t evaluations, each robot could then independently perform the calculation to determine the next policy π and continue with the next iteration. In contrast, Nelder and Mead's Downhill Simplex method requires much stricter control over which policies are evaluated. Thus the robots would need to explicitly coordinate which policies they are to evaluate, and find a way to re-do evaluations that are interrupted by battery changes.

It is important to note that our approach requires a reasonable starting policy. Since we are not explicitly trying to find a stable gait, starting from an unstable gait can lead to mechanical failures that can make it difficult for the Aibos to make much progress. As a result, our method is probably not yet directly applicable to the problem of finding an initial stable walk (e.g. for a bipedal robot).

Since we distributed the learning process over multiple robots, the policies that we discovered were not tuned to any particular robot. While the hardware that the Aibos are made up of is theoretically homogeneous, cumulative wear and tear over time can lead to significant differences between Aibos. It is therefore possible that we could achieve further increases in performance by training individual robots to fine-tune their parameters.

Another interesting avenue for future work is learning fast omni-directional gaits. While the work presented in this paper has focused on maximizing velocity in a forward direction, methods similar to those presented here could be used to optimize turning gaits or gaits for movement in other directions.

Based on our success on the Aibos, we surmise that the learning methods presented in this paper will generalize to other types of legged robots as well. Given a rough gait designed by a human, our algorithm should enable the robot to fine-tune for speed without human intervention.

VI. Conclusion

In this paper, we presented our approach to automatically learning a fast walk on a quadruped robot, namely the Aibo ERS-210A. The policy gradient approach we use allows for distributed, efficient policy evaluation, with all learning happening directly on the robots. Using this method, we have generated one of the fastest known walks on the Aibo: 291 mm/s. Video of our initial gait, training process, and final gait are all available on-line.[7]

VII. Acknowledgments

We would like to thank the members of the UT Austin Villa team, and Daniel Stronger in particular, for their efforts in developing the gaits and software mentioned in this paper. This research was supported in part by NSF CAREER award IIS-0237699.

References

[1] P. Stone, T. Balch, and G. Kraetszchmar, Eds., *RoboCup-2000: Robot Soccer World Cup IV*, ser. Lecture Notes in Artificial Intelligence 2019. Berlin: Springer Verlag, 2001.

[2] J. A. Bagnell and J. Schneider, “Autonomous helicopter control using reinforcement learning policy search methods,” in *International Conference on Robotics and Automation*, 2001.

[3] A. Y. Ng, H. J. Kim, M. I. Jordan, and S. Sastry, “Autonomous helicopter flight via reinforcement learning,” in *Advances in Neural Information Processing Systems 17*. MIT Press, 2004, to Appear.

[4] R. Zhang and P. Vadakkepat, “An evolutionary algorithm for trajectory based gait generation of biped robot,” in *Proceedings of the International Conference on Computational Intelligence, Robotics and Autonomous Systems*, Singapore, 2003.

[5] G. S. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata, “Autonomous evolution of gaits with the Sony quadruped robot,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. 2. Orlando, Florida, USA: Morgan Kaufmann, 13-17 1999, pp. 1297–1304. [Online]. Available: citeseer.nj.nec.com/hornby99autonomous.html

[6] G. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, and M. Fujita, “Evolving robust gaits with AIBO,” in *IEEE International Conference on Robotics and Automation*, 2000, pp. 3040–3045.

[7] M. S. Kim and W. Uther, “Automatic gait optimisation for quadruped robots,” in *Australasian Conference on Robotics and Automation*, Brisbane, December 2003.

[8] M. J. Quinlan, S. K. Chalup, and R. H. Middleton, “Techniques for improving vision and locomotion on the sony aibo robot,” in *Australasian Conference on Robotics and Automation*, Brisbane, December 2003.

[9] T. Rofer, H.-D. Burkhard, U. Duffert, J. Hoffman, D. Gohring, M. Jungel, M. Lotzach, O. v. Stryk, R. Brunn, M. Kallnik, M. Kunz, S. Petters, M. Risler, M. Stelzer, I. Dahm, M. Wachter, K. Engel, A. Osterhues, C. Schumann, and J. Ziegler, “Germanteam robocup 2003,” Tech. Rep., 2003. [Online]. Available: http://www.robocup.de/germanteam/GT2003.pdf

[10] P. Stone, K. Dresner, S. T. Erdoğan, P. Fidelman, N. K. Jong, N. Kohl, G. Kuhlmann, E. Lin, M. Sridharan, D. Stronger, and G. Hariharan, “UT Austin Villa 2003: A new RoboCup four-legged team,” The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, Tech. Rep. UT-AI-TR-03-304, 2003, at http://www.cs.utexas.edu/home/department/pubsforms.shtml.

[11] B. Hengst, D. Ibbotson, S. B. Pham, and C. Sammut, “Omnidirectional motion for quadruped robots,” in *RoboCup International Symposium, Lecture Notes in Computer Science, Lecture Notes in Artificial Intelligence LNAI 2377*, A. Birk, S. Coradeschi, and S. Tadokoro, Eds. Springer, 2001, p. 368.

[12] R. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Advances in Neural Information Processing Systems*, vol. 12. The MIT Press, 2000, pp. 1057–1063.

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

[14] C. J. C. H. Watkins, “Learning from delayed rewards,” Ph.D. dissertation, King's College, Cambridge, UK, 1989.

[15] W. H. Press, *Numerical Recipes in C: the art of scientific computing*. Cambridge: Cambridge University Press, 1988.

[16] J. D. Weingarten, G. A. D. Lopes, M. Buehler, R. E. Groff, and D. E. Koditschek, “Automated gait adaptation for legged robots,” in *Submitted to 2004 IEEE International Conference on Robotics and Automation*, 2003.

[17] D. Stronger and P. Stone, “A model-based approach to robot joint control,” November 2003, under Review. Available from http://www.cs.utexas.edu/˜pstone/papers.html.

[7] `www.cs.utexas.edu/˜AustinVilla/legged/learned-walk/`