Learning Ball Acquisition on a Physical Robot

Peggy Fidelman and Peter Stone
Department of Computer Sciences, The University of Texas at Austin
1 University Station C0500, Austin, Texas 78712-1188
{peggy,pstone}@cs.utexas.edu
http://www.cs.utexas.edu/~{peggy,pstone}

***Abstract*— For a robot to learn to improve its performance based entirely on real-world environmental feedback, the robot's behavior specification and learning algorithm must be constructed so as to enable data-efficient learning. Building upon previous work enabling a quadrupedal robot to learn a fast walk with all of the training done on the physical robot and with no human intervention [1], we demonstrate the ability of the same robot to learn a more high-level, goal-oriented task using the same methodology. In particular, we enable the robot to learn to *capture* (or "grasp") a ball. The learning occurs over about three hours of robot run time and generates a behavior that is significantly better than a baseline hand-coded behavior. Our method is fully implemented and tested on a Sony Aibo ERS-7 robot.**

Keywords: task learning, manipulation and grasping, legged robots

I. Introduction

Recent developments in commercial robotics suggest that before long robots will be ubiquitous in the real world. On the one hand, companies are deploying increasingly affordable consumer robots such as Roomba [2] and Aibo [3]. On the other hand, they are creating increasingly sophisticated humanoid robots such as Asimo [4] and Qrio [5]. As robots do begin to make their way out into the real world, it will become increasingly necessary for them to adapt to constantly changing, unstructured environments [6]. That is, robots must be able to *learn* to improve their performance based entirely on feedback from their physical environment, and as a result, from relatively few training examples.

This paper is concerned with enabling a robot to learn high-level goal-oriented behaviors. Coding these behaviors by hand can be time-consuming, and it often leads to brittle solutions that need to be revised whenever the environment changes or the low-level skills that comprise the behavior are refined. Machine learning offers the promise of a way to generate solutions with little human interaction, so that when the environment changes the solution can be revised with no more than a few hours of machine time. It is also possible for machine learning to lead to better initial solutions than hand-tuning, because humans are often biased toward exploring a small part of the space of possible solutions, whereas machine learning explores the space in a systematic way.

Particular challenges arise when there is no simulator available for the robot. Because each trial requires interaction with the physical world in real time, it is not possible to offset the costs of an inefficient learning algorithm with a faster processor. The learning algorithm absolutely must make efficient use of the information gained from each trial.

Previous work has demonstrated that it is possible to learn locomotion entirely autonomously, with no human intervention other than battery changes [1], [7]. The success of the work done by Kohl and Stone was due to several features of the task being learned, namely:

- The ability of the robot to get its own reward signal;
- The possibility of expressing the policy for this task with a set of parameters; and
- The existence of an efficient algorithm for learning to optimize such a parameterized policy.

The main contribution of this paper is to extend this methodology to a higher-level, more goal-oriented behavior, namely ball acquisition by a soccer-playing Aibo robot. Although previous work has demonstrated the benefits of using machine learning in conjunction with higher-level behaviors, that learning did not serve to *improve* the behavior but rather to predict the effects of the behavior, which was hand-coded [8].

The remainder of this paper is organized as follows. Section II describes the background and motivation for this work. Section III specifies the task to be learned and its parameterization. Section IV describes the primary machine learning algorithm used in the work, as well as the setup of the training scenario carried out by the robot. Section V details the results of the training, and Section VI discusses the contributions of this work, as well as possible directions for the future.

II. Background

Acquiring an object is a prerequisite for many types of manipulations in the world [9], [10]. For example, in the case of a Sony Aibo robot playing soccer, one of our motivating test-bed domains [11], it is much easier to design effective ways for the robot to kick the ball if we may assume that the ball starts in a specific position relative to the robot. Furthermore, if the robot can grasp the ball securely enough, it can move the ball by turning with the ball until the ball reaches a field position from which the kick is likely to move the ball to the desired destination (such as in the opponent's goal). Thus, as a representative task, we consider the goal of having a robot walk up to a ball and gain control of it. For the purposes of this paper, we define *control* to mean that the

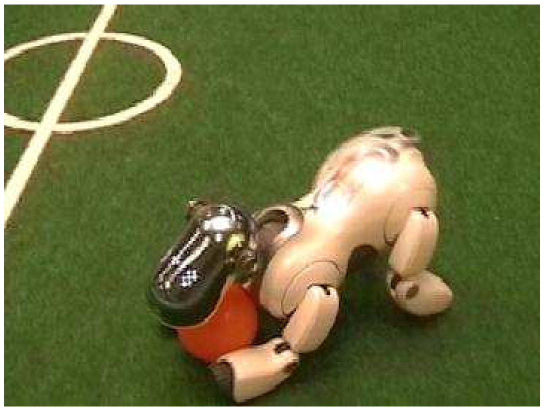
Fig. 1. An Aibo with control of the ball.

robot holds the ball under its chin in such a way that it can turn with the ball as shown in Figure 1.

As the robot platform for this research, we use the commercially available Sony Aibo ERS-7, a quadruped robot [3]. The ERS-7 has a head with three degrees of freedom, and a CMOS camera in the head. It has several pressure sensors and two infrared range sensors, one on the head and one on the chest. The robot is able to capture frames from the camera at a rate of 25 Hz, from which our software recognizes objects, such as the orange ball, based on color segmentation and aggregation [12]. The infrared chest sensor is relatively noisy, but provides enough resolution to reliably detect whether or not the ball is indeed captured as shown in Figure 1 (note that in this capture position, the ball is not within range of the camera, which resides in the tip of the Aibo's nose). This variety of sensors allows us to rely only on local sensing. In addition, the 576 MHz 64 bit RISC processor allows all necessary processing to be done onboard. In this work, we use a system for vision processing, walking, and kicking that was developed as part of our larger robot soccer project [11].

III. Task Specification

The task learned in this paper is motivated by our ongoing development of the UT Austin Villa four-legged robot soccer team [11]. Our team adopted the following strategy for getting the ball into the under-chin position described above: when the Aibo is walking to a ball with the intent of kicking it and gets "close enough," it first slows down to allow for more precise positioning, and then it lowers its head to capture the ball under its chin (this is the "capturing motion").

However, executing this motion so that the ball is not knocked away in the process is a challenge: if the head is lowered when the ball is too far away, the head may knock the ball away, but if it is not lowered in time, the body of the robot may bump the ball away. Naturally, the perceived ball distance at which the head should be lowered depends on how fast the robot is walking, so the amount that the robot slows down when close to the ball must be tuned simultaneously with the timing of the capturing motion. For the same reason, every time the speed of the base gait changes (due to a newly-designed gait or a different walking surface, for example), the approach must be re-tuned. This is a time-consuming task to perform by hand.

The parameters that control the transition from walking to capturing the ball are as follows:

- `slowdown_dist`: the ball distance (in millimeters) at which slowing down begins;
- `slowdown_factor`: the (multiplicative) factor, in the range [0,1], by which the gait slows down at this point;
- `capture_angle`: the maximum ball angle (in degrees) at which the capturing motion may begin (see Figure 2);
- `capture_dist`: the ball distance (in millimeters) at which the capturing motion begins (if the ball is within the specified angle).

IV. Machine Learning Methodology

Given this parameterization, we are faced with a parameter optimization problem in four dimensions. Because our policies can be expressed in this way, and because our domain has the same efficiency constraints as that of learning fast locomotion for the Aibo, the policy gradient reinforcement learning algorithm used by Kohl and Stone [13] is a natural choice.

By this method, starting from a base policy $\{\theta_1, ..., \theta_N\}$, $t-1$ new policies are chosen by selecting one of $\{\theta_i - \epsilon_i, \theta_i, \theta_i + \epsilon_i\}$ randomly for each dimension i, where ϵ_i is a fixed increment particular to dimension i. These t policies (the base policy and the $t-1$ randomly selected policies) are then evaluated, and their scores are used to estimate the partial derivative in each of the N dimensions, which leads to a new base policy.

The estimation of partial derivatives works as follows. For each dimension i, the policies are divided into three sets according to the value of parameter i: if its value is $\theta_i - \epsilon_i$, the policy is in set $S_{-\epsilon,i}$; if it is θ_i, the policy is in set $S_{0,i}$; and if it is $\theta_i + \epsilon_i$, the policy is in set $S_{+\epsilon,i}$. Then the average score over all the policies in each set is computed and used to build an adjustment vector A of size N. For each i, if the average score over the set $S_{0,i}$ is greater than the average score over each of the other two sets, then $A_i = 0$; otherwise, A_i becomes the difference between the average scores over set $S_{+\epsilon,i}$ and set $S_{-\epsilon,i}$. A is then normalized and multiplied by a scalar step size η, so that we will adjust our policy by a fixed amount each time. The above process comprises one iteration of the algorithm. For the parameters used in our work, see Table I.

For comparison, we also implemented and tested the other two algorithms with which Kohl and Stone had some success in learning locomotion [1]. One of these algorithms, hill climbing, is identical to the policy gradient algorithm except for the way the new base policy is chosen. Rather than using the scores from the current iteration to calculate which direction we should move in policy space, in hill climbing the base policy for the next iteration is simply set equal to

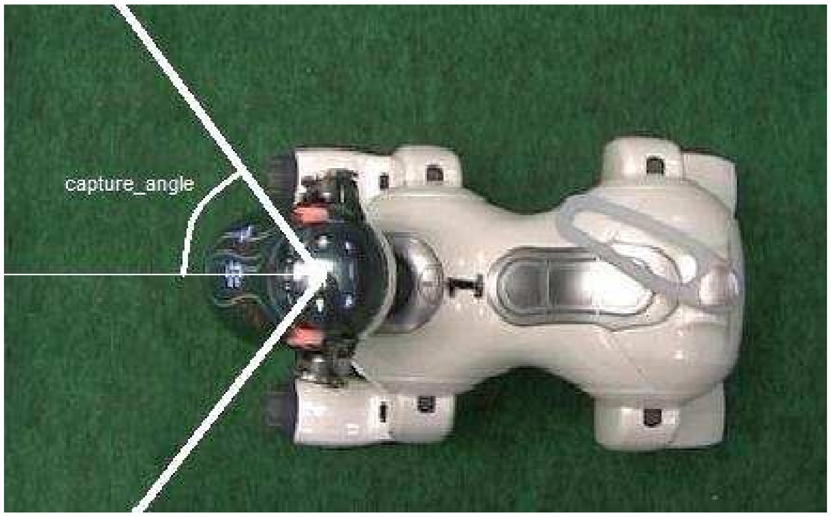

Fig. 2. Illustration of `capture_angle`. If the Aibo believes that the center of the ball is to the right of the thick white lines, then it will continue to turn toward the ball rather than beginning the capturing motion, even if the ball distance is believed to be less than `capture_dist`.

TABLE I
PARAMETERS FOR THE POLICY GRADIENT ALGORITHM

Parameter	Value
Policies per iteration (t)	8
Increment for `slowdown_dist` (ϵ_1)	20
Increment for `slowdown_factor` (ϵ_2)	0.5
Increment for `capture_angle` (ϵ_3)	5
Increment for `capture_dist` (ϵ_4)	20
Scalar step size (η)	1

the policy that got the best score in the current iteration. The other algorithm, known as the downhill simplex or "amoeba" algorithm, involves moving a simplex of points through policy space [14]. The simplex undergoes various transformations in response to the scores received by the policies represented by its vertices. These transformations are intended to move the simplex toward more promising parts of the policy space while preserving its volume. Unlike the policy gradient and hill climbing algorithms, the amoeba algorithm can effectively adjust its own step size based on the information it is receiving about the fitness of the policies that make up its vertices. More details about the hill climbing and amoeba algorithms can be found in previous work [1].

One particular challenge for learning ball acquisition is defining an appropriate reward signal. In learning locomotion, the time that each potential walk takes to traverse a fixed distance generates a feedback signal that can rank the relative merits of each policy on a continuous scale. In our case, however, there is no straightforward way to rate a particular policy with regard to "how well" it captures the ball: it either does or it does not.

Therefore, we use a binary reinforcement signal: if the robot captures the ball, it receives a reward of 1; if not, it receives a reward of 0. The Aibo can determine autonomously whether it has captured the ball by means of the infrared range sensor on its chest, which reliably registers noise over a certain level when the ball is being held under the chin. During training, the score for a given policy is determined by running 12 trials with that policy and averaging the reinforcement signal over those trials (thus effectively producing a discrete, ordinal reinforcement signal).

Each trial consists of the robot approaching the ball from a random location on the standard field used in RoboCup, which is surrounded by a short wall designed to keep the ball from leaving the field. If the Aibo successfully captures the ball, it kicks it in whichever direction it estimates is away from the wall. (This is to ensure that we will get enough trials with the ball in the open field, since our kicks are powerful enough that the ball is often stopped by the wall if kicked from less than halfway across the field. Trials that begin with the ball at a wall are also less informative, since capturing the ball here is much harder, and even a good policy will fail a lot more along the wall, which can lead to a smaller spread of scores among policies.) Then, before starting the next trial, the Aibo turns around approximately $180°$ in place in order to knock the ball away from it if it is still close. Once it has done this, it begins the next trial by searching for the ball and then approaching it with the parameters of the current policy (see Figure 3)[1].

All learning is done on the Aibo itself, including all calculations necessary to execute the learning algorithm. Interruptions caused by dead batteries are of little consequence, since the algorithm has practically no state: if we resume from the last base policy, we will never lose as much as an entire iteration.

[1]Videos depicting the training process can be found at: `http://www.cs.utexas.edu/~AustinVilla/legged/learned-acquisition/`

```
base policy ← initial policy
while base policy is still improving do
  $policies[0]$ ← base policy
  for $i \in [1, t-1]$ do
    $policies[i]$ ← new random policy
  end for
  for $i \in [0, t-1]$ do
    $totalscore \leftarrow 0$
    $slowdown\_dist \leftarrow policies[i][1]$
    $slowdown\_factor \leftarrow policies[i][2]$
    $capture\_angle \leftarrow policies[i][3]$
    $capture\_dist \leftarrow policies[i][4]$
    for $j \in [1, num\_trials\_per\_policy]$ do
      while ball farther than slowdown_dist do
        walk to ball at maxspeed
      end while
      while ball farther than $capture\_dist$ and outside of
      $capture\_angle$ do
        walk to ball at speed of $(slowdown\_factor) * maxspeed$
      end while
      lower head over ball
      if chest IR sensor senses ball then
        $totalscore \leftarrow totalscore + 1$
        if center of field to robot's left then
          kick to left
        else
          kick to right
        end if
      end if
      turn $180°$
    end for
    $pscore[i] \leftarrow totalscore/(num\_trials\_per\_policy)$
  end for
  UPDATE base policy
end while
```

Fig. 3. Algorithm for learning to approach the ball. In the case of the policy gradient algorithm, the UPDATE routine chooses a new base policy according to an estimation of the gradient of policy space. In the case of hill climbing, UPDATE sets the base policy to be the highest-scoring policy from the last iteration. The amoeba algorithm does not have "iterations" per se, but each policy $policies[i]$ is evaluated in the same way as shown here.

V. Results

Starting from a hand-tuned[2] policy with an acquisition success rate of roughly 36%, the robot reached a policy[3] with a success rate of approximately 64% within seven iterations, which takes approximately three hours (see Figure 4). With 12 trials for each policy, and 8 policies per iteration, roughly[4] 672 trials were needed.

This result demonstrates the ability of our robots to learn this task. Hill climbing and the amoeba algorithm had success comparable to that of the policy gradient algorithm (see Figure 5).

Table II shows the values of the best policies found by the learning algorithms. Note that all methods learned not to slow down at all (`slowdown_factor`=1). When `slowdown_factor` is 1, the parameter `slowdown_dist` has no effect on the robot's behavior, which is presumably why the different learning methods produced such a wide range of values for this parameter.

The fact that all methods learn not to slow down is a demonstration of the advantage that machine learning can bestow because of its systematic exploration of the space: in hand-tuning, we believed that slowing down would make the ball approach more reliable at the expense of speed, since the estimations of ball distance should change less rapidly if the robot is walking more slowly. But we can see here that this is actually not the case, since our system which optimized only for reliability actually found that slowing down at all is only a disadvantage.

It is worth noting that for the policy gradient and amoeba algorithms, the results we present here correspond to the *first* sets of learning parameters we tried (and for the hill climbing algorithm, the results correspond to the second set we tried). Consequently, we suspect that these methods are not particularly sensitive to their parameter settings. In any case, these experiments demonstrate the potential reduction in hand-tuning that can be achieved via machine learning methods.

VI. Discussion and Conclusion

The results in Section V demonstrate that we are able to achieve significant improvements on the ball-capturing task via autonomous machine learning. All of the learning is done on a physical robot, with no human intervention, over the course of a relatively short training time. The resulting training paradigm is useful in that it saves us time, and can generate better policies, when compared to manual tuning of an approach behavior. Indeed, we are using the described automated training paradigm in our competitive team development for the RoboCup 2004 robot soccer competition.

The fact that several different learning algorithms generate comparable improvements from the initial hand-tuned policy suggests that the main reason for our learning success is the setup of the task. The results themselves are not very sensitive to the details of the learning approach (neither to the algorithm used nor to the parameters used for that algorithm). Thus, the main contribution of this paper is the demonstration that a non-

[2]This policy was actually hand-tuned for a different parameterization of the space and then converted mathematically to fit the new parameterization. However, preliminary work with the original parameterization of the space, for which the base policy was meticulously tuned, showed that learning generated a policy which improved over the hand-tuned one by an amount comparable to the improvement we see here in the new parameter space.

[3]Videos of the initial policy and the best learned policy can be found at: `http://www.cs.utexas.edu/~AustinVilla/legged/learned-acquisition/`

[4]This is approximate because it does not take lost trials due to dead batteries into account.

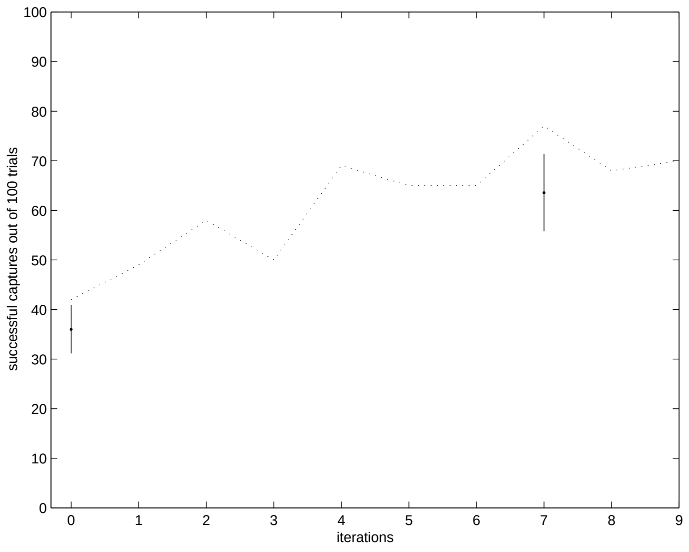

Fig. 4. Progress of the policy gradient algorithm. The dotted line shows the learning curve, produced by running 100-trial evaluations on the base policy of each iteration. Error bars (showing the 95% confidence interval) are displayed for the two policies (initial and best learned) which were later evaluated with seven 100-trial runs to establish statistical significance. The fact that the later evaluations yield significantly lower scores is most likely due to robot deterioration over time.

TABLE II
POLICY VALUES LEARNED BY EACH ALGORITHM.

Policy	slowdown_dist	slowdown_factor	capture_angle	capture_dist
Initial	200	0.7	15	110
Best: policy gradient	125.4	1	17.4	152.4
Best: amoeba algorithm	208.4	1	33.4	161.7
Best: hill climbing	240	1	35	170

trivial goal-oriented behavior can be learned efficiently on a mobile robot with no human intervention.

In our ongoing research, we aim to identify additional behaviors that can be learned in a similarly autonomous and efficient fashion, and ultimately to characterize the full range of characteristics of a task required to enable a mobile robot to learn to improve its task performance by our methods.

ACKNOWLEDGMENTS

Thanks to the members of the UT Austin Villa team for their efforts in developing the software used as a basis for the work reported in this paper. Special thanks to Nate Kohl for sharing his machine learning infrastructure used for Aibo locomotion. This research was supported in part by NSF CAREER award IIS-0237699 and ONR YIP award N00014-04-1-0545.

REFERENCES

[1] N. Kohl and P. Stone, "Machine learning for fast quadrupedal locomotion," in *The Nineteenth National Conference on Artificial Intelligence*, July 2004.
[2] iRobot, "roomba vacuum cleaner robot," http://www.roombavac.com/.
[3] Sony, "Aibo robot," 2004, http://www.sony.net/Products/aibo.
[4] Honda, "Asimo robot," http://world.honda.com/ASIMO/.
[5] Sony, "Qrio robot," http://www.sony.net/SonyInfo/QRIO/.
[6] R. A. Brooks and M. J. Mataric, "Real robots, real learning problems," in *Robot Learning*, J. H. Connell and S. Mahadevan, Eds. Kluwer Academic Publishers, 1993, pp. 193–213.
[7] T. Roefer, "Evolutionary gait-optimization using a fitness function based on proprioception," in *RoboCup-2004: Robot Soccer World Cup VIII*, D. Nardi, M. Riedmiller, and C. Sammut, Eds. Berlin: Springer Verlag, 2005, to appear.
[8] S. Chernova and M. Veloso, "Learning and using models of kicking motions for legged robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2004.
[9] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2000. [Online]. Available: http://citeseer.ist.psu.edu/bicchi00robotic.html
[10] I. Kamon, T. Flash, and S. Edelman, "Learning to grasp using visual information," The Weizmann Institute of Science, Revhovot, Israel, Tech. Rep., March 1994. [Online]. Available: http://citeseer.ist.psu.edu/kamon94learning.html
[11] 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, "The UT Austin Villa 2003 four-legged team," in *RoboCup-2003: Robot Soccer World Cup VII*, D. Polani, B. Browning, A. Bonarini, and K. Yoshida, Eds. Berlin: Springer Verlag, 2004.
[12] 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

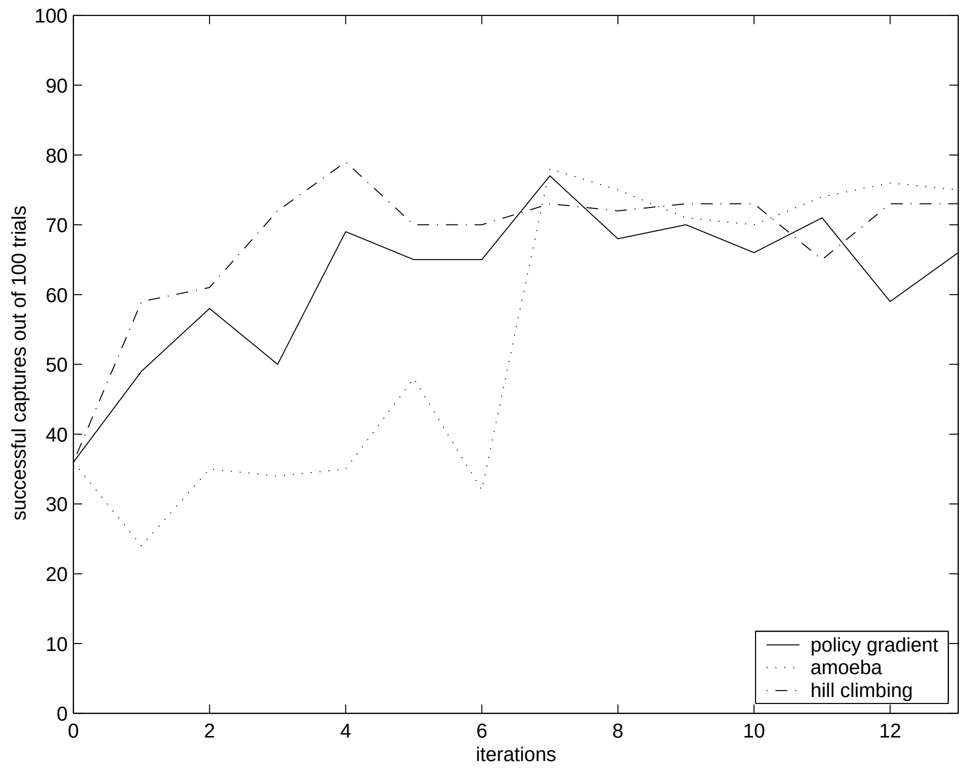

Fig. 5. Progress of the amoeba algorithm and hill climbing, in comparison to the progress of the policy gradient algorithm. The learning curves for the policy gradient algorithm and hill climbing were produced by running 100-trial evaluations on the base policy of each iteration. For the amoeba algorithm, since there is no concept of a "base policy," 100-trial evaluations were run on every eighth policy tried by the algorithm (because with hill climbing and the policy gradient algorithm, eight policies were tried per iteration). The robot that we had been using for all training failed after iteration 9 of the policy gradient algorithm, so iterations 10 through 13 of the policy gradient algorithm were carried out on a different robot than the rest.

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/ftp/pub/AI-Lab/index/html/Abstracts.2003.html#%03-304.

[13] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2004.

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