The Chin Pinch: A Case Study in Skill Learning on a Legged Robot

Peggy Fidelman and Peter Stone

Department of Computer Sciences, The University of Texas at Austin
1 University Station C0500, Austin, Texas 78712-0233
{peggy,pstone}@cs.utexas.edu

Abstract. When developing skills on a physical robot, it is appealing to turn to modern machine learning methods in order to automate the process. However, when no accurate simulator exists for the type of motion in question, all learning must occur on the physical robot itself. In such a case, there is a high premium on quick, efficient learning (specifically, learning with low sample complexity). Recent results in learning locomotion have demonstrated the feasibility of learning fast walks directly on quadrupedal robots. This paper demonstrates that it is also possible to learn a higher-level skill requiring more fine motor coordination, again with all learning occurring directly on the robot. In particular, the paper presents a learned ball-grasping skill on a commercially available Sony Aibo robot, with no human intervention other than battery changes. The learned skill significantly outperforms our best hand-tuned solution. As the learned grasping skill relies on a learned walk, we characterize our learning implementation within the layered learning formalism. To our knowledge, the two learned layers represent the first use of layered learning on a physical robot.

Keywords: learning and adaptive systems, sensor-motor control.

1 Introduction

In order for robots to be useful for many real-world applications, they must be able to adapt to novel and changing environments. Ideally, a robot should be able to respond to a change in its surroundings by adapting both its low-level skills, such as its walking style, and the higher-level skills which depend on them. Because hand-coding is time-consuming and often leads to brittle solutions, this adaptation should occur as autonomously as possible. Machine learning promises a way to generate solutions with little human interaction, so that when the environment changes the solution can be revised with limited human effort. Machine learning can also lead to better 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.

Current learning methods typically need a large amount of training data to be effective. Thus, an appealing approach to creating learning *robots* is to train behaviors first in simulation before implementing them in the real world [5].

However, especially when concerned with complex perception or manipulation tasks, we cannot assume an adequate simulator will always exist for a given robot. With no simulator, each trial requires interaction with the physical world in real time. In such cases, it is not possible to offset the costs of an inefficient learning algorithm with a faster processor. The learning algorithm must make efficient use of the information gained from each trial (i.e., it must have low sample complexity).

For this reason, until recently, most of the locomotion approaches for quadrupedal robots have centered around hand-tuning a parameterized gait. However, in recent years, there has been a spate of research on efficient learning algorithms for quadrupedal locomotion [2,4,9,11,12,13,14]. A common feature of these approaches is that the robots time themselves walking across a known, fixed distance, thus eliminating the need for any human supervision.

This paper demonstrates that it is possible to similarly learn a higher-level more fine-motor skill, again with all learning occurring directly on the robot. In particular, the paper presents a learned ball-grasping skill on a commercially available Sony Aibo robot, with no human intervention other than battery changes. We show that a learning algorithm that has proven effective for learning walks applies directly to this new task. However, due to the different task characteristics, significant changes to the training scenario are required. This paper contributes a full specification of a training scenario that enables autonomous learning of a ball-grasping skill. The learned skill significantly outperforms an extensively hand-tuned solution.

As the learned grasping skill relies on a learned walk itself, we characterize our learning implementation within the layered learning formalism. *Layered learning* [17] is a hierarchical machine learning paradigm that leverages a given task decomposition to learn complex tasks efficiently. A key feature is that the learning of each subtask directly facilitates the learning of the next-higher subtask layer. Layered learning has been used previously to generate complex, multi-layer behaviors in *simulated* environments [6,7,17,18]. To our knowledge, our two learned layers represent the first use of layered learning on a physical robot.

The remainder of this paper is organized as follows. Section 2 describes the background and motivation for this work. Section 3 specifies the tasks to be learned and how the layered learning paradigm can be used to relate them, as well as how the training scenario is set up for each task. Section 4 describes the primary machine learning algorithm used in the work. Section 5 details the results of the training, and Section 6 discusses the contributions of this work, as well as possible directions for the future.

2 Background

This section describes the robot hardware used in all experiments and introduces the target task towards which it is trained (Section 2.1). It also summarizes the layered learning formalism (Section 2.2) within which we frame our approach.

2.1 Ball Acquisition by a Legged Robot

Acquiring an object is a prerequisite for many types of manipulations in the world [1,8]. For example, in the case of a Sony Aibo robot playing soccer, one of our motivating testbed domains, 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 into a better position relative to the objects in the robot's environment before executing a kick. (For example, the robot can turn with the ball until it is pointed at the opponent's goal.) Thus, as a representative high-level task for learning, we consider the aim 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 robot holds the ball under its chin in a way that allows it to 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 [15]. The ERS-7 has four legs with three degrees of freedom in each, a head with three degrees of freedom, and a CMOS camera in the head. It has several pressure sensors and two infrared range sensors, as well as position sensors in each of its joints. The robot is able to capture frames from the camera at a rate of 30 Hz. From these images, our software recognizes objects such as the orange ball based on color segmentation and aggregation. This variety of sensors allows us to rely on local sensing alone. 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 [16].

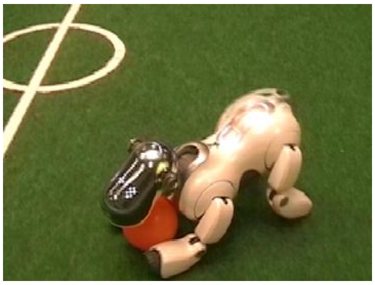
Fig. 1. An Aibo with control of a ball. Achieving this position without knocking the ball away in the process is a challenge; our learning method allows the Aibo to do this more reliably without sacrificing walking speed.

2.2 Layered Learning

Layered learning is a general hierarchical machine learning paradigm that leverages a given task decomposition to learn complex tasks efficiently. Though it has been validated previously in simulation, this paper presents the first application of layered learning on a physical robot. Specifically, the robot first learns a fast walk, then uses that walk to approach the ball while learning to grasp it.

The main principles of layered learning are summarized in Table 1. A detailed description of these principles is given by Stone and Veloso[17].

We cast our learned behaviors within the formal layered learning framework as defined in the remainder of this section [17]. Consider the learning task of identifying a hypothesis h from among a class of hypotheses H which map a set of

Table 1. The key principles of layered learning

1. Learning a mapping directly from inputs to outputs is not tractable.
2. A bottom-up, hierarchical task decomposition is given.
3. Machine learning exploits data to train and/or adapt. Learning occurs separately at each level.
4. The output of learning in one layer feeds into the next layer.

state feature variables S to a set of outputs O such that, based on a set of training examples, h is most likely (of the hypotheses in H) to represent unseen examples.

When using the layered learning paradigm, the complete learning task is decomposed into hierarchical subtask layers $\{L_1, L_2, \ldots, L_n\}$ with each layer defined as

$$L_i = (\boldsymbol{F_i}, O_i, T_i, M_i, h_i)$$

where:

$\boldsymbol{F_i}$ is the input vector of state features relevant for learning subtask L_i. $\boldsymbol{F_i} = <F_i^1, F_i^2, \ldots>$. $\forall j,\ F_1^j \in S$.

$\boldsymbol{O_i}$ is the set of outputs from among which to choose for subtask L_i. $O_n = O$.

$\boldsymbol{T_i}$ is the set of training examples used for learning subtask L_i. Each element of T_i consists of a correspondence between an input feature vector $\boldsymbol{f} \in \boldsymbol{F_i}$ and $o \in O_i$.

$\boldsymbol{M_i}$ is the ML algorithm used at layer L_i to select a hypothesis mapping $\boldsymbol{F_i} \mapsto O_i$ based on T_i.

$\boldsymbol{h_i}$ is the result of running M_i on T_i. h_i is a function from $\boldsymbol{F_i}$ to O_i.

Note that a layer describes more than a subtask; it also describes an approach to solving that subtask and the resulting solution.

As stated in the Decomposition principle of layered learning, the definitions of the layers L_i are given *a priori*. The Interaction principle is addressed as follows. $\forall i < n$, h_i directly affects L_{i+1} in at least one of three ways:

– h_i is used to construct one or more features F_{i+1}^k.
– h_i is used to construct elements of T_{i+1}; and/or
– h_i is used to prune the output set O_{i+1}.

It is noted above in the definition of $\boldsymbol{F_i}$ that $\forall j,\ F_1^j \in S$. Since $\boldsymbol{F_{i+1}}$ can consist of new features constructed using h_i, the more general version of the above special case is that $\forall i, j,\ F_i^j \in S \cup_{k=1}^{i-1} O_k$.

When training a particular component, layered learning freezes the components trained in previous layers, thereby adding additional constraints to the learning process. It also adds guidance, by training each layer in a special environment intended to prepare it well for the target domain.

The original implementation of the layered learning paradigm was on the full robot soccer task in the RoboCup soccer simulator [17]. First, a neural network was used to learn an interception behavior. This behavior was used to train a decision tree for pass evaluation, which was in turn used to generate the input representation for a reinforcement learning approach to pass selection.

A subsequent application of layered learning uses two layers, each learned via genetic programming, for a soccer keepaway task in a simplified abstraction of the TeamBots environment [7]. In the full TeamBots environment, four learned layers were used, also on a keepaway task [18]. To our knowledge, there has been no previous implementation of layered learning on a physical robot.

3 Layered Learning on a Physical Robot

The process of approaching a ball and then gaining control of it relies on the gait that allows the robot to move toward the ball. Thus, when both the gait and the grasping are individually learned, we have a layered learning hierarchy consisting of two layers. This section casts the recent research on gait learning within the layered learning formalism (L_1), and then builds upon it to learn ball grasping, a second, higher-level skill (L_2).

3.1 Learning a Gait

In recent years, several approaches to learning a gait on an Aibo have been studied. Among these approaches, most of the differences between gaits stem from the shape of the loci through which the feet pass and the exact parameterizations of those loci. For example, Kohl and Stone used elliptical loci to learn high-speed walks using a policy gradient learning approach [11], while simultaneously but independently, Quinlan et al. were able to generate high-velocity gaits using a genetic algorithm and loci of arbitrary shape [12], and Roefer created a flexible gait implementation that allows use of a variety of different shapes of loci [14]. They then used an evolutionary learning algorithm to optimize a novel fitness function based on proprioception to learn a fast gait [13]. Chernova and Veloso similarly used an evolutionary approach with good success [2] and Lee et al. refined Kohl and Stone's approach to estimate gait speeds more effectively [4].

This paper builds upon the successful approach of Kohl and Stone [11], in which the gait is defined by a set of 12 continuous parameters specifying, among other things, the shape of the trajectory through which each leg moves as well as the target heights of the front and rear of the body. Thus, gait learning is framed as a parameter optimization problem, with forward speed as the objective function. The learning is accomplished via the policy gradient algorithm summarized in Section 4.

The fitness of a policy, or set of values for the 12 parameters, is obtained by having one or more Aibos time themselves as they walk a fixed, known distance indicated by a pair of landmarks. To reduce the effect of noise, this evaluation process is performed three times for each policy, and the resulting times are averaged to get the fitness of the policy.

In the notation of layered learning, the gait layer (L_1) is thus defined as:

$\boldsymbol{F_1}$: $\emptyset$;
O_1: values for the 12 parameters defining a gait, plus the speed of the resulting gait;

T_1: the set of training examples obtained by recording the time it takes to walk back and forth across a fixed distance;
M_1: the policy gradient algorithm described in Section 4;
h_1: the parameters of the fastest discovered gait, and its speed.

The walks learned using this technique perform similarly to those reported by Stone and Kohl [11], who report that with three robots continually walking across the field more than 1000 total times for approximately 3 hours, they achieved the fastest known walk on the Aibo at the time. Notably, the robots learned without any human intervention other than battery changes approximately once an hour, and the walk speed was nearly doubled during training.

Though our learned walk itself is a reproduction of previous results, the formulation of the walking task within the framework of layered learning is novel to this paper. Next, Section 3.2 introduces this novel learned skill in full detail and similarly frames it within layered learning. As prescribed by layered learning, the new skill uses the learned walk (h_1) as a part of its training scenario.

3.2 Learning to Acquire the Ball

The task of learning to capture a ball under the robot's chin is motivated by the ongoing development of our four-legged robot soccer team [16]. The robot is only able to kick in certain directions, so it is useful to be able to capture the ball and turn with it before kicking. Our team adopted the following strategy for getting the ball into this position: 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 (the *capturing motion*).

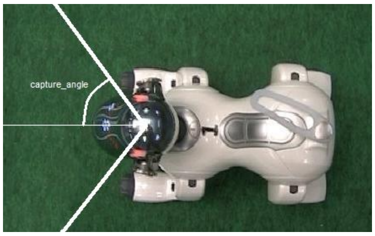

Fig. 2. Illustration of `capture_angle`. If the Aibo believes that the center of the ball is to the right of the thick white lines, then it will continue to turn toward the ball rather than beginning the capturing motion, even if the ball distance is believed to be less than `capture_dist`.

Executing the capturing motion without knocking the ball away 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. Furthermore, certain aspects of the acquisition motion interact, such as the perceived ball distance at which the head should be lowered and the amount that the robot slows down when close to the ball. Parameters like these must therefore be tuned simultaneously. This entire process is time-consuming to perform by hand.

The parameters that control the transition from walking to capturing the ball, as indicated in Figure 3, 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);
– `turn_cutoff`: the minimum ball angle (in degrees) at which the robot will not move directly toward the ball at all, but instead will turn in place to face the ball more directly. This parameter controls how straight the final part of the robot's approach will be.

```
totalscore ← 0
for j ∈ [1, n] do
  locate ball
  while ball farther than slowdown_dist do
    if ball angle more than turn_cutoff then
      turn toward ball
    else
      walk to ball at maxspeed
    end if
  end while
  while ball farther than capture_dist and outside of capture_angle do
    if ball angle more than turn_cutoff then
      turn toward ball
    else
      walk to ball at maxspeed*slowdown_factor
    end if
  end while
  lower head over ball
  if head tilt position sensor senses ball then
    totalscore ← 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
policy_score ← totalscore/n
```

Fig. 3. Method for evaluating policies while learning to approach the ball. n is the number of trials per policy; in our experiments, we used $n = 12$.

Given this parameterization, we are faced with a parameter optimization problem in five 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 learning algorithm used to learn the gait (see Section 4) is again a natural choice.

However, there are new challenges in learning ball acquisition; specifically, i) defining an appropriate reward signal, and ii) defining an appropriate training scenario. The policy gradient algorithm relies on the magnitude of the fitness difference between policies. This magnitude is readily available in the learned walking scenario, because speed provides a natural and continuous measure of fitness. But in the case of ball acquisition, 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 trying to put its chin all the way down to its chest and then taking note of the value of the position sensor in its head tilt joint; if the ball is indeed under its chin, the head tilt motor will stop moving before getting to the requested position. During training, the score for a given policy is determined by running a fixed number of trials (12) with that policy and averaging the reinforcement signal over those trials (thus producing a discrete reinforcement signal). In other words, a policy's score is the number of times it successfully captures the ball over the course of 12 trials: an integer between 0 and 12 inclusive.

Each trial consists of the robot approaching the ball from a random location on the standard field used in the 2004 RoboCup competition, which is surrounded by a short wall designed to keep the ball from leaving the field. The training procedure is summarized in pseudocode in Figure 3.

One goal of the training procedure is to generate as many trials as possible in the open field, rather than with the ball starting against the wall. The latter trials are somewhat less informative because capturing the ball along the wall is considerably harder; even a good policy will fail much more frequently along the wall, which can lead to a smaller spread of scores among policies. In order to keep the ball in the open field, if the Aibo successfully captures it, it kicks it in whichever direction it estimates is away from the wall (lines 21–25 in Figure 3). 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 (line 27), so as to make the different trials as independent as possible. 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 (lines 3–17). Videos depicting the training process in action are available online[1].

In the notation of layered learning, we thus have the following definition of the acquisition layer (L_2):

- $\boldsymbol{F_2}$: {BallAngle, BallDistance} $\in \{[-180, 180], [0, \infty)\}$. The five thresholds that comprise an acquisition policy (`slowdown_dist`, etc.) relate to these two sensor readings alone;
- O_2: whether or not to lower the head at the current time;
- T_2: evaluations of mappings from $\boldsymbol{F_i}$ to O_i, obtained by repeatedly trying to grasp the ball by the process described above and summarized in Figure 3. In particular, the learned walk (h_1) is used during training;
- M_2: the policy gradient algorithm described in Section 4;
- h_2: the final learned acquisition policy.

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 learning algorithm we use has practically no state: if we resume from its last base policy, we will never lose as much as an entire iteration of the algorithm. With the algorithm parameters used in our experiments, a battery typically lasts for the amount of time necessary to complete two iterations, so on average a run requires about 4 battery changes.

[1] `http://www.cs.utexas.edu/~AustinVilla/legged/learned-acquisition/`

4 The Policy Gradient Algorithm

The learning algorithm common to both learned layers estimates the gradient of the policy's value function near the current policy via efficient experimentation. It then takes a step in the direction of the estimated gradient and repeats the process. We use the policy gradient algorithm presented and evaluated against alternatives for learned locomotion by Kohl and Stone [10]. This section summarizes the algorithm in task-independent terms and points out some of its advantages for the purpose of ball acquisition.

Table 2. Parameters for the policy gradient algorithm in the ball acquisition learning task

Parameter	Value
Policies per iteration (t)	8
Increment for `slowdown_dist` (ϵ_1)	10mm
Increment for `slowdown_factor` (ϵ_2)	0.1
Increment for `capture_angle` (ϵ_3)	5°
Increment for `capture_dist` (ϵ_4)	10mm
Increment for `turn_cutoff` (ϵ_5)	10°
Scalar step size (η)	2

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 for their fitness. Their scores are used to estimate the partial derivative of fitness with respect to 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 is 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 the policy is adjusted by a fixed amount each time. The above process comprises one iteration of the algorithm. For the parameters used in learning ball acquisition, see Table 2.

5 Results

The success of layer L_1 at producing a significantly faster forward gait has been demonstrated previously [11]. In this paper, we demonstrate that, in the layered learning paradigm we present here, L_2 can build upon the gait improvement conferred by L_1. In particular, we hypothesize the ability to learn a significantly improved ball-acquisition skill to go with the significantly improved gait.

To test this hypothesis quantitatively, we learn ball acquisition using three gaits learned by separate runs of layer L_1. All three of these learned gaits represent significant improvements in speed over the initial hand-tuned gait. The initial (before learning) ball acquisition policy was hand-tuned for the initial hand-tuned gait from which gait learning began. The ball-acquisition learning paradigm described by layer L_2 was then applied to each of these gaits, and significantly improved acquisition policies were discovered for all three.

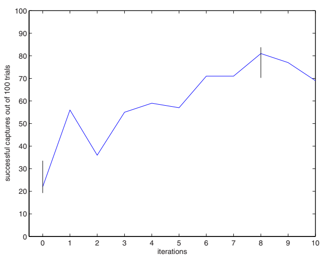

Fig. 4. Progress of acquisition learning on gait A. This learning curve was produced by running 100-trial evaluations on the base policy of each iteration. Error bars (showing the 95% confidence interval) are depicted for the initial and best learned policy; these were obtained by running five 100-trial evaluations on each policy.

Figure 4 shows the learning curve for one of these gaits, which we will refer to as gait A. For this gait, the initial ball acquisition policy acquires the ball roughly 26% of the time, whereas the best learned policy acquires the ball approximately 77% of the time. This improvement was reached in 8 iterations, which requires 768 attempted acquisitions (approximately 3 hours). The initial policy and the best learned policy were each subjected to five 100-trial evaluations, resulting in five approximations of the success rate of each. Statistical significance was then established by applying a t-test to these success rates.

Gait A has a speed of about 315mm/sec, whereas the initial hand-tuned gait from which it was learned has a speed of about 245mm/sec. The gait training process also requires roughly 3 hours. Therefore, with 6 hours of training, our robot's walking speed increased 29% and its reliability at acquiring the ball more than doubled[2] in comparison with the original hand-coded solution.

Table 3 summarizes the ball acquisition policies learned for all three gaits. It also shows the success rate of each when tested on the gait with which it was learned. These success rates were obtained by running 100-trial evaluations of the policy (except for gait A, where the data is the result of all 500 trials run to establish statistical significance on the data in Figure 4). The success rate of the initial hand-coded policy is 26% for gait A, 14% for gait B, and 14% for gait C.

Note that in all cases, the method learned not to slow down at all (`slowdown_factor` is 1). When `slowdown_factor` is 1, the parameter `slowdown_dist` has no effect on the robot's behavior, which is presumably why learning resulted in such a wide range of values for this parameter.

The fact that in all cases our method learns not to slow down demonstrates the advantage that machine learning can bestow because of its unbiased 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 estimates of ball distance should change less rapidly if the robot is walking more slowly. Our system, however – which optimized only for reliability – found that slowing

[2] The initial ball-acquisition skill had a success rate of 36% with the initial gait, and was the result of extensive tuning involving the testing of dozens of parameter settings over the course of several days.

down is in fact a disadvantage: in all learning trials it actively increased the `slowdown_factor` parameter from its initial value of 0.8 to 1.0.

Table 3. Policy values learned for each gait, and the approximate success rate of each

Policy	slowdown_dist	slowdown_factor	capture_angle	capture_dist	turn_cutoff	**Success rate**
Initial	200	0.8	15	110	90	26%/14%/14%
Best: gait A	193	1	32	155	80	77%
Best: gait B	187	1	19	155	69	84%
Best: gait C	228	1	31	129	84	52%

We originally hypothesized that different gaits would require different acquisition policies. This hypothesis was supported by the fact that the initial ball acquisition policy dropped in effectiveness from approximately 36% on the gait for which it was hand-tuned to 14–26% on the learned gaits.

However, it turned out not to be the case with these learned gaits and their trained acquisition policies. Rather, upon testing the best acquisition policy learned with gait A on each of the other two learned gaits, there was no significant difference in performance — if anything, the acquisition policy learned on gait A performs better in each case, as shown in Table 4.

Table 4. Success rates of best natively learned acquisition policy and best acquisition policy learned on gait A

Gait	Natively learned	Best on gait A
B	84%	91%
C	52%	53%

Nonetheless, the layered learning paradigm enabled the separation of the learning for the walk and ball acquisition into two distinct phases. Given that they learn most efficiently in different training environments, such a hierarchical approach is an essential component of our successful skill learning.

6 Conclusion

This paper makes two main contributions: i) a significantly improved grasping skill achieved via fully autonomous machine learning with all training and computation executed on-board the robot, and ii) the first instantiation of layered learning on a physical robot.

The layered learning approach to locomotion and ball acquisition learning that we describe here is very useful in practice. Compared to manually tuning these skills, this method saves time and can generate better policies. Indeed, we used the described automated training paradigms for both the gait and the acquisition in our competitive team development for the RoboCup 2004 and 2005 robot soccer competitions, reaching the semifinals (out of 8) at the regional event and the quarterfinals (out of 24) at the international event both years [16].

In our ongoing research, we aim to identify additional skills and behaviors that can be learned in a similarly autonomous and efficient fashion. Several candidates for an L_3 that builds on the grasping skill learned in L_2 exist. Currently, for example, all design and tuning of kicks for our RoboCup team are done by hand. If this process could be automated, it would likely save time and might also lead to improved solutions. However, since most kicks begin by grasping the ball, autonomous learning of kicks would be intractable without a good grasping behavior. Another possible candidate for an L_3 that builds on the learned grasping skill is the tuning of walks that manipulate the ball, such as the one used in the turning-with-ball behavior which makes grasping so crucial in the first place (see Section 2.1). Eventually, these learned skills may feed into still higher-level learned decision-making behaviors (where to pass or when to shoot) based on the current learned skills. Indeed, an immediately realizable L_3 related to kicking is the *modeling* of hand-tuned kicks as accomplished via regression learning by Chernova and Veloso [3]. They use these models to demonstrably improve the robot's decision-making when choosing form among different kicks. Ultimately, we hope to characterize the full range of characteristics of tasks on a mobile robot that may be improved by these methods.

Acknowledgments

Thanks to Uli Grasemann and Bikram Banerjee for valuable comments, and also 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, ONR YIP award N00014-04-1-0545, and DARPA grant HR0011-04-1-0035.

References

1. Bicchi, A., Kumar, V.: Robotic grasping and contact: A review. In: Proceedings of the IEEE International Conference on Robotics and Automation (April 2000)
2. Chernova, S., Veloso, M.: An evolutionary approach to gait learning for four-legged robots. In: Proceedings of IROS'04 (September 2004)
3. Chernova, S., Veloso, M.: Learning and using models of kicking motions for legged robots. In: Proceedings of International Conference on Robotics and Automation (ICRA'04) (May 2004)
4. Cohen, D., Ooi, Y.H., Vernaza, P., Lee, D.D.: The University of Pennsylvania RoboCup 2004 legged soccer team. Available at URL `http://www.cis.upenn.edu/robocup/UPenn04.pdf`
5. Gat, E.: On the role of simulation in the study of autonomous mobile robots. In: AAAI-95 Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents. Stanford, CA (March 1995)
6. Gustafson, S.M.: Layered learning for a cooperative robot soccer problem. Master's thesis, Kansas State University (2000)

7. Hsu, W.H., Gustafson, S.M.: Genetic programming and multi-agent layered learning by reinforcements. In: Genetic and Evolutionary Computation Conference, New York,NY, pp. 764–771. Morgan Kaufmann, San Francisco (2002)
8. Kamon, I., Flash, T., Edelman, S.: Learning to grasp using visual information. Technical report, The Weizmann Institute of Science, Revhovot, Israel (March 1994)
9. Kim, M.S., Uther, W.: Automatic gait optimisation for quadruped robots. In: Australasian Conference on Robotics and Automation, Brisbane (December 2003)
10. Kohl, N., Stone, P.: Machine learning for fast quadrupedal locomotion. In: The Nineteenth National Conference on Artificial Intelligence, pp. 611–616 (July 2004)
11. Kohl, N., Stone, P.: Policy gradient reinforcement learning for fast quadrupedal locomotion. In: Proceedings of the IEEE International Conference on Robotics and Automation (May 2004)
12. Quinlan, M.J., Chalup, S.K., Middleton, R.H.: Techniques for improving vision and locomotion on the sony aibo robot. In: Proceedings of the 2003 Australasian Conference on Robotics and Automation (December 2003)
13. Röfer, T.: Evolutionary gait-optimization using a fitness function based on proprioception. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, Springer, Heidelberg (2005)
14. Rofer, T., Burkhard, H.-D., Duffert, U., Hoffman, J., Gohring, D., Jungel, M., Lotzach, M., Stryk, O.v., Brunn, R., Kallnik, M., Kunz, M., Petters, S., Risler, M., Stelzer, M., Dahm, I., Wachter, M., Engel, K., Osterhues, A., Schumann, C., Ziegler, J.: Germanteam robocup 2003. Technical report (2003)
15. Sony: Aibo robot (2004) `http://www.sony.net/Products/aibo`
16. Stone, P., Dresner, K., Fidelman, P., Jong, N.K., Kohl, N., Kuhlmann, G., Sridharan, M., Stronger, D.: The UT Austin Villa 2004 RoboCup four-legged team: Coming of age. Technical Report UT-AI-TR-04-313, The University of Texas at Austin, Department of Computer Sciences, AI Laboratory (October 2004)
17. Stone, P., Veloso, M.: Layered learning. In: López de Mántaras, R., Plaza, E. (eds.) ECML 2000. LNCS (LNAI), vol. 1810, pp. 369–381. Springer, Heidelberg (2000)
18. Whiteson, S., Kohl, N., Miikkulainen, R., Stone, P.: Evolving keepaway soccer players through task decomposition. Machine Learning 59(1), 5–30 (2005)