Integrated Learning for Interactive Synthetic Characters

Bruce Blumberg, Marc Downie, Yuri Ivanov, Matt Berlin, Michael Patrick Johnson, Bill Tomlinson

Synthetic Characters Group, The Media Lab, MIT *

Abstract

The ability to learn is a potentially compelling and important quality for interactive synthetic characters. To that end, we describe a practical approach to real-time learning for synthetic characters. Our implementation is grounded in the techniques of reinforcement learning and informed by insights from animal training. It simplifies the learning task for characters by (a) enabling them to take advantage of predictable regularities in their world, (b) allowing them to make maximal use of any supervisory signals, and (c) making them easy to train by humans.

We built an autonomous animated dog that can be trained with a technique used to train real dogs called "clicker training". Capabilities demonstrated include being trained to recognize and use acoustic patterns as cues for actions, as well as to synthesize new actions from novel paths through its motion space.

A key contribution of this paper is to demonstrate that by addressing the three problems of state, action, and state-action space discovery at the same time, the solution for each becomes easier. Finally, we articulate heuristics and design principles that make learning practical for synthetic characters.

CR Categories: I.2.6 [Artificial Intelligence]: Learning—Concept Learning; I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods and Search—Heuristic Methods; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques

Keywords: behavioral animation, animation, computer games

1 Introduction

We believe that interactive synthetic characters must learn from experience if they are to be compelling over extended periods of time. Furthermore, they must adapt in ways that are immediately understandable, important and ultimately meaningful to the people interacting with them. Nature provides an excellent example of systems that do just this: pets such as dogs.

Remarkably, dogs do this with minimal insight into our behavior, and little understanding of words and gestures beyond their use as cues. In addition, dogs are only able to learn causality if the events, actions and consequences are proximate in space and time, and as long as the consequences are motivationally significant. Nonetheless, the learning dogs do allow them to behave commonsensically and ultimately exploit the highly adaptive niche of "man's best friend." Our belief is that by embedding the kind of learning of which dogs are capable into synthetic characters, we can provide them with an equally robust mechanism for adapting in meaningful ways to the people with whom they are interacting.

In this paper, we describe a practical approach to real-time learning for synthetic characters that allows them to learn the kinds of things that dogs seem to learn so easily. We ground our work in the traditional techniques of reinforcement learning, in which a creature learns to maximize reward in the absence of a teacher. Additionally, our approach is informed by insights from animal training, where a teacher is available. Animals and their trainers act as a coupled system to guide the animal's exploration of its state, action, and state-action spaces (see Section 3.2). Therefore, we can simplify the learning task for autonomous animated characters by (a) enabling them to take advantage of predictable regularities in their world, (b) allowing them to make maximal use of any supervisory signals, either explicit or implicit, that the world offers, and (c) making them easy to train by humans.

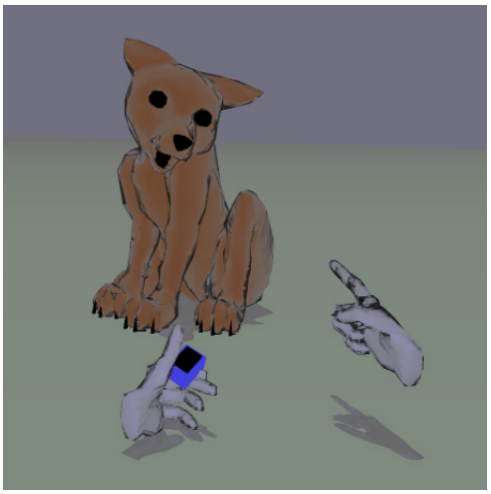

Figure 1: Terence is an autonomous animated pup that can be trained using clicker training. The trainer's interface is a microphone and pair of virtual hands controlled by a gamepad. The left hand holds a clicker that makes a sound when pressed. The right hand serves as a target for luring, and can also give extra reward by scratching the dog's head.

Using this system, we implemented the autonomous animated dog shown in Figure 1 that can be trained with a technique used to train real dogs. The synthetic dog thus mimics some of a real dog's ability to learn including:

- The best action to perform in a given context.
- What form of a given action is most reliable in producing reward.
- The relative reliability of its actions in producing a reward and altering its choice of action accordingly.

*email:bruce,marcd,yivanov,mattb,aries,badger@media.mit.edu

- To recognize new and valuable contexts such as acoustic patterns.
- To synthesize new actions by being "lured" into novel configurations or trajectories by the trainer.

In order to accomplish these learning tasks, the system must address the three important problems of state, action and state-action space discovery. A key contribution of this paper is to show how these processes may be addressed in an integrated approach that guides and simplifies the individual processes.

We emphasize that our behavioral architecture is one in which learning can occur, rather than an architecture that solely performs learning. As we will see, learning has important implications for many aspects of a general behavior architecture, from the design of the perceptual mechanism to the design of the motor system. Conversely, careful attention to the design of these components can dramatically facilitate the learning process. Hence, an important goal of this paper is to highlight some of these key design considerations and to provide useful insights apart from the specifics of the approach that we have taken.

We begin by surveying related work and putting our work in perspective. We then turn to a discussion of reinforcement learning. We introduce the core concepts and terminology, discuss why a naïve application of reinforcement learning to synthetic characters is problematic, and finally draw on insights from animal training on how animals conceptually address the same issues. We then describe our approach, reviewing our key representations and processes for state, action and state-action space discovery. We present our experience with Terence, our virtual pup, and discuss limitations of our approach. We conclude with a summary of what we see as the key lessons from our work.

2 Related Work

The approach described below is in the general category of reinforcement learning. (See [Kaelbling 1990; Ballard 1997; Mitchell 1997; Sutton and Barto 1998] for good introductions to the field.) The incremental exploration of state-action space proposed below is similar to an approach originally suggested by Drescher [1991]. What is new in our work is an integrated approach to state, action and state-action space discovery within the context of reinforcement learning and an articulation of heuristics and design principles that make learning practical for synthetic characters.

Our approach is also informed by a close study of animal training and what it seems to imply about how animals learn. For good introductions to animal learning, see [Lorenz and Leyahusen 1973; Lorenz 1981; Shettleworth 1998; Gould and Gould 1999; Gallistel and Gibbon 2000; Lindsay 2000; Coppinger and Coppinger 2001]; for an introduction to the field of animal training see [Ramirez 1999]; and for an introduction to the specific approach to training that we take as our inspiration, i.e., "clicker training", see [Wilkes 1995; Pryor 1999; Ramirez 1999]. Clicker training has been successfully adapted by researchers at SONY CSL to train their robotic dog AIBO [Kaplan et al. 2001]. See [Yoon et al. 2000a] for an early application of clicker training to training animated characters. What is novel in our research is a computational model that not only uses animal training as a starting point, but places learning within the larger behavioral context.

In an effort to reduce the work required by animators, learning has been applied to the problem of generating motion primitives. (See [van de Panne and Fiume. 1993; van de Panne et al. 1994; Grzeszczuk and Terzopoulos 1995; Grzeszczuk et al. 1998; Hodgins and Pollard 1997; Gleicher 1998].) Most recently, [Faloutsos et al. 2001] have done exciting work showing how a statistical learning technique (SVM) can be used to learn the "pre-conditions" from which a given "specialist controller" can succeed at its task, thus allowing them to be combined into a general purpose motor system for physically based animated characters.

The approaches to motor learning described above focus on learning "how to move" subject to some criteria such as energy minimization, whereas the motor learning that is described in this paper focuses on learning the "value with respect to a motivational goal of moving in a certain way". As such, our approach represents a layer above many of these prior approaches. Finally we note our emphasis is on learning as an online capability to enhance interaction with a human participant rather than as a design tool.

A number of noteworthy architectures for control of animated autonomous characters have been proposed [Reynolds 1987; Tu and Terzopoulos 1994; Blumberg and Gaylean 1995; Perlin and Goldberg 1996; Funge et al. 1999; Burke et al. 2001]. While producing impressive results, most of these systems have not incorporated behavioral learning and thus cannot modify the pre-specified behavior on the basis of experience. Our contribution is to integrate learning into a general-purpose behavior architecture.

Higher-level behavioral learning has only begun to be explored in computer graphics. (For examples, see [Yoon et al. 2000b; Burke et al. 2001; Tomlinson and Blumberg 2002].) Several of the current generations of digital pets such as *Dogz* [Resner et al. 1997], *Creatures* [Grand et al. 1996] and AIBO also incorporate simple learning. This is done particularly well in *Dogz*, to the point that many people are convinced that more learning is going on than is actually the case. Factors contributing to this assumption include: immediate emotional responses by the creature to good or bad consequences, intuitive means for delivering reward or punishment, and an immediate and noticeable change in behavior in response. The popular video game *Black and White* [Evans 2002] centrally features a character that learns from a person's actions. Our contribution is to provide insights into how state and action space discovery can be integrated into the learning process.

3 Background on Learning and Training

The approach taken in our work is best understood as a variant of a popular machine learning technique known as reinforcement learning. In this section we begin by introducing the key ideas and terminology. We then look at the problem from the perspective of animal training and highlight the key ideas from animal training that can help make reinforcement learning practical for interactive synthetic characters.

3.1 Introduction to Reinforcement Learning

Reinforcement learning (RL) is often used by autonomous systems that must learn from experience. In reinforcement learning, the world in which the creature lives is assumed to be in one of a set of perceivable states. The goal of reinforcement learning is to learn an optimal sequence of actions that will take the creature from an arbitrary state to a goal state in which it receives a reward. The main approach taken by reinforcement learning is to probabilistically explore states, actions and their outcomes to learn how to act in any given situation. Before we describe how this is done, we need to define state, action and reward a bit more formally.

State refers to a specific, hopefully useful, configuration of the world as sensed by the creature's entire sensory system. As such, state can be thought of as a label that is assigned to a sensed configuration. The space of all represented configurations of the world is known as the *state space*.

Performing an *action* is how a creature can affect the state of its world. Typically, the creature is assumed to have a finite set of actions, from which it can perform exactly one at any given instant,

e.g., walk or eat. The set of all possible actions is referred to as the *action space*.

A *state-action* pair, denoted as <*S*/*A*>, is a relationship between a state *S* and an action *A*. It is typically accompanied by some numeric value, e.g., *future expected reward*, that indicates how much benefit there is in taking the action *A* when the creature senses state *S*. Based on this relationship a *policy* is built, which represents a probability with which the creature selects an action given a specific state.

The creature receives *reinforcement* (or *reward*) when it reaches a state in which it can satisfy a goal. For example, if a dog sits and gets a treat for doing so, the reward or reinforcement is the resulting decrease in hunger or pleasure in eating the treat.

Credit assignment is the process of updating the associated value of a state-action pair to reflect its apparent utility for ultimately receiving reward.

While there are a number of variants of reinforcement learning, *Q-Learning* is a simple and popular representative that can be used to illustrate some key concepts. In Q-Learning, introduced by Watkins [Watkins and Dayan 1992], the state-action space is discretized if necessary and stored in a lookup table. In the table, each row represents a state, and each column represents an action. An entry in the table represents the "utility", or *Q-Value,* of a given state-action pair with respect to getting a reward. Watkins showed that the optimal value for each state-action pair could be learned by incrementally (and exhaustively) exploring the space of state-action pairs and by using a local update rule to reflect the consequences of taking a given action in a given state with respect to achieving the goal state [Sutton 1991].

It is important to note that techniques such as Q-Learning that focus on learning an optimal sequence of actions to get to a goal state solve a much harder problem than either animals solve or that we need to solve for synthetic characters. As we will see, animals are biased to learn proximate causality. Even in the case of sequences, the noted ethologist Leyahusen suggests that the individual actions may be largely self-reinforcing, rather than being reinforced via back propagation [Lorenz and Leyahusen 1973]. In addition, Nature places a premium on learning adequate solutions quickly.

Reinforcement learning is an example of an unsupervised learning technique in that the only supervisory signal is the reward received when it achieves a goal. On the other hand, it is clear that a trainer could significantly expedite exploration of the respective spaces by guiding the search. In the following section we discuss how trainers and their animals cooperate to simplify the learning task.

3.2 The Perspective of Animal Training

Here we describe a popular and easy technique for animal training called "clicker training" and what it seems to imply about how animals learn.

Clicker training unfolds in three basic steps. The first step is to create an association between the sound of a toy clicker and a food reward. A dog conditioned to the clicker will expectantly look for a treat upon hearing the click sound. Once the association between clicks and treats is made, trainers use the click sound to "mark" behaviors that they wish to encourage. By clicking when the dog performs a desired behavior, and subsequently treating, the dog begins to perform the behavior more frequently.

Animals appear to make an important simplifying assumption: an action or stimulus that immediately precedes a motivationally significant consequence is "as good as causal." Hence, clicker training is a particularly effective training technique because it makes it easy to provide immediate feedback. Indeed, the sound of the clicker marks the exact behavior that leads to the subsequent treat, as well as signaling that the action is complete. In addition, it acts as a bridge between when the dog *earned* the reward and when it actually *receives* it.

Since clicker training relies on the dog to produce some approximation of a desired behavior before it can be rewarded (and producing a high level of reinforcement keeps the dog interested in the process), trainers utilize a variety of techniques to encourage the dog to perform behaviors they might otherwise perform infrequently, or not at all. A useful and popular technique is to train the dog to touch an object such as the trainer's hand or a "target stick". By subsequently manipulating the position of the target, the trainer can, in effect, *lure* the dog through a trajectory or into a pose as it follows its nose. For example, by moving the target over the dog's head, a dog may be lured into sitting down. If lured and rewarded repeatedly, the dog will begin to produce the action (e.g., sit) without being lured. This suggests that the animal is associating reward with its resulting body configuration or trajectory, and not for the action of simply following its nose.

The dog is unlikely to perform the desired final form of the behavior immediately, especially if it is an unusual behavior, e.g., "dancing on the two rear feet". As a result, the trainer will often guide the dog toward the desired behavior by rewarding ever-closer approximations in a process known as *shaping*.

The third and final step in clicker training is to add a discriminative stimulus such as a gesture or vocal cue. Trainers typically introduce the cue by presenting it as the animal is just beginning to perform the action, and then subsequently rewarding the action. Significantly, the animal *has already decided what to do before the trainer issues a cue* but is still able to learn to associate the action (and its subsequent reward) with a cue occurring in a temporal window proximate to the action onset. Note, unlike other training techniques, clicker trainers teach the action first, and then the cue. The superiority of this decomposition suggests that animals make associations more easily if they already "know" a particular action is valuable.

3.3 Making Learning Practical for Synthetic Characters

While reinforcement learning provides a theoretically sound basis for building systems that learn, there are a number of issues that make it problematic in the context of autonomous animated creatures. Borrowing ideas from animal training, however, we can address these problems in a way that makes real-time learning practical for synthetic characters.

Enable them to take advantage of predictable regularities in their world. We saw that dogs use predictable regularities of how the world works to simplify the learning task. For example, they bias their choice of action toward those actions that have been successful at receiving reward in the past. Similarly, they limit their attention to stimuli or cues that occur in a temporal window around an action's onset in order to identify reliable contexts in which to perform the action. Through variations in how the action is performed and by attending to correlations between the action's reliability in producing reward and the state of contemporaneous stimuli, they are performing a local search in a potentially valuable neighborhood.

This model of causality, while very simple, is nonetheless sufficient to capture many aspects of how the world works. Perhaps as important for synthetic characters, learning proximate causality is exactly the kind of learning that is most apparent and easiest to understand for an observer. A final insight is that the state and action spaces often contain a natural hierarchical organization that facilitates the search process.

Allow them to make maximal use of any supervisory signals, either explicit or implicit, that the world offers. Biasing the

choice of behavior based on consequences is an example of making use of *explicit* supervisory signals (such as getting a treat). The consequences of the action can also be used as an *implicit* (secondary) supervisory signal for guiding the exploration of the character's state and action spaces. This guidance is significant because synthetic characters, by their very nature, have state and action spaces that are both continuous and far too big to permit an exhaustive search, even if discretized. For example, the *a priori* state space for a character that must learn to respond to arbitrary verbal or gestural cues, will be intractably huge since it will include the entire set of possible acoustic and gestural patterns. Similarly, in the case of an expressive character for whom the style of the action is as important as the action itself, the action space will be the space of all possible motions. Ironically though, most of the volume of these respective spaces is irrelevant from the character's standpoint of getting reward.

Our observation from animal training is that animals seem to solve this problem by building models of important sensory cues "on demand", using rewarded actions as the context for identifying important sensory cues and for guiding the perceptual model of the cue. For example, a good example of the acoustic pattern "sit" is the one that occurs just before or during a sit action that results in reward.

This point suggests a computational strategy—discover, based on experience, those patterns (in the case of state space) or motions (in the case of action space) that *do* seem to matter and add them dynamically to their respective spaces. These processes are known as *state space discovery* and *action space discovery* respectively. While there are established techniques for performing state-space discovery (see, for example, [Ivanov 2001]) they often require a lot of data. A key insight is that these processes can be guided by using the context of a rewarded action to facilitate the classification process. Indeed, by choosing the right representation, state and action space discovery can be done using exactly the same mechanism.

Make them easy to train. For training to be a compelling experience for the human participant, the character needs to be easy to train using observable behavior, without the trainer having any visibility into the character's internal state.

On the simplest level, the character must be sensitive to the immediate consequences of its actions, attend to changes in stimuli that occur right before and during its performance of an action, and its observable behavior must change quickly in response. The ability to be trained via luring is especially important since otherwise the trainer has to wait for the animal to randomly choose the action, which could take forever.

Our discussion of animal training suggests that animals perform the equivalent of credit assignment in a way that makes it easier to train them than it might be otherwise. In the case of luring, they generalize from being rewarded for "following their nose" to being rewarded for their resulting configuration or trajectory. In the language of reinforcement learning, it is as if during credit assignment the "follow your nose" state-action pair lets *another state-action pair get the credit*, namely the one associated with the configuration or trajectory. Similarly, when associating a cue with an action, animals act as if they form and assign credit to *new* state-action pairs based on evidence acquired while performing an existing but related state-action pair (i.e., one that shares the same action). The computational implication of luring and cue association is that by allowing the state-action pair that would normally get credit to delegate its credit to another pair, the training process can be facilitated.

4 System Description

We now turn to a moderately detailed discussion of the representations and processes used in our system. While due to space we cannot provide all the implementation details, we try to offer insights in design choices. The system described below has been implemented as part of a system that is similar to that described in [Burke et al. 2001; Isla et al. 2001].

4.1 Key Representations

4.1.1 State

Many state spaces have a natural hierarchical organization, e.g., the space of acoustic patterns, the space of utterances, and individual utterances such as "sit", "down" and "roll over". By incorporating a similar hierarchical representation of state space into our system, we can "notice" that a given action is more reliable when a whole *class* of states is active. This information provides evidence that further exploration and refinement within a class of states might be fruitful for increasing reliability of reward.

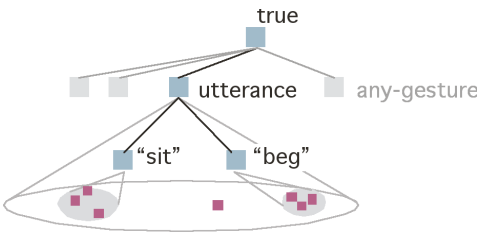

Figure 2: In our work the state space is represented by a percept tree. The percept tree maintains a hierarchical representation of the sensory input where leaf nodes represent the highest degree of specialization and the root node matches any sensory input. The structure of the tree is sequentially discovered and refined with time as indicated by its utility with respect to getting reward.

As illustrated in Figure 2, we use a hierarchical mechanism called a *percept tree* to extract state information from the world. Each node in the tree is called a *percept,* with more specific percepts nearer to the leaves. Percepts are atomic perception units, with arbitrarily complex logic, whose job it is to recognize and extract features from raw sensory data. For example, one percept may recognize the presence of the utterance "sit" in an auditory stream, and another might recognize the performance of a particular motor trajectory. Similarly, an "utterance" percept might recognize the presence of "utterances" in an auditory field, and its children might recognize the presence of specific utterances such as "sit", "down", "roll-over", etc. The root of the tree is the most general percept, which we call "True" since it is always active.

Percepts are model-based recognizers, meaning that on each simulation cycle they compare raw sensory data to an internal model and become *active* if they match within some threshold. If a percept is active, the sensory data is passed recursively to the percept's children for more specific classification. If not, all its children can be pruned from the update cycle. This culling is important since percept models can vary in complexity. For symbolic data, the model is trivial: it is a string and the matching criterion is simple string equality. In the case of an utterance percept, however, the model may be a collection of vectors of cepstral coefficients[Rabiner and Juang 1993] that represent the mean of a set of previously learned examples [Ivanov 2001], and the comparison between sensory data and the model is more complex (section 4.2.3). Motion percepts use a model that represents a path through the space of possible motions. Also associated with each percept is a short-term memory mechanism that keeps track of its activation history over some period of time.

In the language of RL, a percept represents a subset of the entire state space. That is, it looks for a specific feature in the state space. In RL, state refers to the *entire* sensed configuration of the world; a

percept is focused on only *one* aspect of that configuration. As we will see, percept decomposition of state allows for a heuristic search through potentially intractable state and state-action spaces. The downside is that it makes learning conjunctions of features harder.

It is important to note that the percept tree is a dynamic structure that is modified as a result of state space discovery as described in Section 4.2.3.

4.1.2 Action

Actions refer to identifiable patterns of motion through time. They are often conceptualized and implemented as discrete verbs, perhaps parameterized with associated adverbs (see [Rose et al. 1999]). While this approach has the desirable property that other parts of the system can treat the action as a label, the representation is not amenable to the type of action space discovery needed to support luring. In contrast, if we consider a creature as having a *pose space* that contains all of its possible body configurations, then an action can be thought of as a specific path through pose space. Just as a percept is a label for a class of observations, an action can be thought of as a label associated with a path or class of paths in pose space. For the purposes of learning, the analogy to state learning is complete if one assumes the existence of a distance metric that evaluates the similarity of two paths. This is the fundamental representation of action used by our system.

Each creature in our system has a motor system with a representation of the creature's pose space encoded in a structure called a *pose-graph*. The nodes in the pose-graph represent annotated configurations that are generated originally from source animation material. A node includes a complete set of joint angles and velocities as well as a number of annotations including time and source-labeling (i.e., what animation it came from and at what point within the animation), connectivity information (e.g., the preceding and following poses in the source animation), and over time, a distribution of the likelihood of being in the current pose as a result of all known actions. For example, a pose associated with a sitting configuration might be the result of sitting or shaking a paw but is unlikely to be associated with being told to jump.

The nodes of this graph are connected together in tangled *directed, weighted graphs*. By associating a distance metric between poses, paths taking the body from pose to pose can be efficiently found and animations can be reformed in real-time by interpolating through nodes together again as needed. Details regarding the actual metric may be found in [Downie 2000] but essentially it captures the intuition that transitioning between similar joint configurations should be preferred over widely differing joint configurations, and that transitions that require less acceleration should be favored over those that require more. Because the pose-graph is derived from "correct" examples, it implicitly captures, to some approximation, many of the biological and physical constraints of how the creature moves—at the very least we are always interpolating within the convex hull of these "correct" examples.

In addition to the pose-graph, the motor system contains *motor programs* that are capable of generating paths through pose-graphs in response to requests from actions. These programs may be quite simple (essentially no more than playing out a particular animation) or more complex (for example, luring towards an object).

One branch of the percept tree is devoted to motor percepts that recognize paths taken by the motor system through pose space. That is, a given motor percept has a model of a path and the capability to compare a novel path to this model. As we will see in section 4.2.4 this allows us to treat action space discovery using almost the identical mechanism as used in state space discovery.

The key points about action are that (a) our underlying representation of action is that of a path through a space of body configurations, (b) we can calculate a distance metric between paths that reflects the similarity between two paths, (c) associated with each path is a "label" and (d) the label is used to specify which path through pose space the motor system should follow at any given point in time.

4.1.3 State-Action

The representation of a particular state-action pair in our system is called an *action tuple*. An action tuple is composed of five elements that specify: what to do, when, to what, for how long, and why. However, one can think of an action tuple as an augmented state-action pair in which the state information is provided by an associated percept (when), and the action (what) is the label for a given path through pose space. Action tuples are organized into groups and compete probabilistically for activation based on their value and applicability (i.e., if their associated percept is active). In the discussion below, we will use action tuple and percept-action pair interchangeably.[1]

Each action tuple keeps reliability and novelty statistics for its associated percept and the percept's children. Reliability models the correlation between an action tuple being rewarded and a percept being active (in an overlapping temporal window). The novelty statistic reflects the relative frequency of the event of the percept being active; a novel percept is one has been rarely active. These statistics are used by the system to guide the exploration of potentially useful states by identifying more specific percepts that seem correlated with an increased reliability of the action in producing reward.

Mirroring our hierarchical representation of state, action tuples that invoke the same action but that depend on different percepts are organized hierarchically according to the specificity of the percept. When a transition between active actions occurs, we perform credit assignment and the outgoing action chooses its "best" action tuple to receive credit. For this approach to work, we need a metric to determine the "best" candidate for credit assignment. This need not be the percept-action pair that actually performed the action. Instead, we find the percept-action pair with the same action, but with a percept that was not only active, but also the most reliable, novel and specific. We search for this pair within a temporal window overlapping with the action performed by some specified amount. This is illustrated in Figure 3. Similarly, during action selection, each action gets to choose its "best" action tuple to compete with the "best" action tuples associated with other actions.

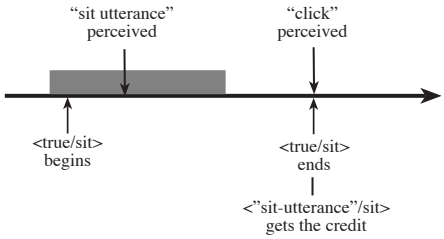

Figure 3: In this example, the <"true"/sit> action tuple delegates credit to <"sit-utterance"/sit> since the "sit-utterance" percept became active during the attention window (gray bar) associated with <"true"/sit> and is a more novel and reliable predictor of reward than "true". By allowing the credit assignment phase to choose who gets credit we can dramatically simplify the learning and training process, as we will see in the section on action space discovery.

[1] We use percept-action pair rather than state-action pair to remind the reader that an action tuple makes its "when decision" based on a subset of the entire state of the world as indicated by its "when" percept.

4.1.4 Reward

An action tuple may have good, indifferent or bad consequences. Consequences are expressed on an absolute scale, and certain events are labeled *a priori* as being "good" or "bad".

4.2 Key Processes

4.2.1 Credit Assignment

Our approach to credit assignment varies from the traditional RL approach in a number of ways:

- **Delegate credit assignment**. The action tuple that is deactivating and normally the candidate for credit assignment has the option to delegate credit assignment to another action tuple. This is perhaps the most significant difference and plays an important role in our algorithm.
- **Selective propagation of value**. The key implication of the bias to learn immediate consequences is that we do not propagate value unless a good or bad consequence is observed, or unless the novelty of the percept associated with the succeeding action tuple is above a threshold. The intuition is that the percept-action pair should only get credit if it produced reward or if it seems causal in making a novel percept active, thereby allowing another potentially more valuable percept-action pair to become active.
- **A rate-based model**. In traditional RL, the scalar value of a state-action pair tends towards the average value of performing that action in that state. An action tuple, on the other hand, explicitly learns a model of its *rate* of producing reward; ultimately, its value is a function of this learned rate and the value assigned to the consequences. During credit assignment, an action tuple updates a model of its rate of producing reward based on consequence.
- **Non-stationary estimate**. The rate of producing a significant consequence is estimated over the most recent N trials, where N is typically a small number. Should the world change, a creature can rapidly update its rate estimates and adapt to the changes. Trials are measured in the number of activations of the action tuple that led to a reward. Hence, they are variable in length, reflecting the pattern of rewards.

The most important reason for using a rate-based model is that by maintaining an explicit model of rate, the action tuple is able to inform the rest of the system whether a consequence is consistent with its model or not, and hence expected or unexpected. For example, this information can be used by a proto-emotion system to decide whether the creature should show surprise or not, and if so, whether the surprise should be positive or negative.

4.2.2 State-Action Space Discovery

State-action space discovery is the process of discovering the best percept-action pair to perform in any given state. In our earlier discussion of RL, we saw that the set of state-action pairs is typically specified *a priori* and the task for the learning algorithm is to exhaustively explore the space and learn the appropriate value for each pair. Our hierarchical representation of state allows us to adopt a different approach—the system is initially populated with only a few percept-action pairs (i.e., action tuples) that represent general world states (i.e., reference percepts at the top of the percept tree). Over time, new percept-action pairs are added as the system gathers evidence that a promising action associated with a given state might be made even more reliable if associated with a more specific child of the state. This process of creating new children action tuples is referred to as *specialization*. At the same time, of course, the system must learn the appropriate value for the percept-action pairs. The advantage of this approach is twofold. First, the system only explores areas of the space for which there is evidence of possible improvement. Second, fewer resources are required when action tuples are not created *a priori*. In this section, we discuss how specialization occurs.

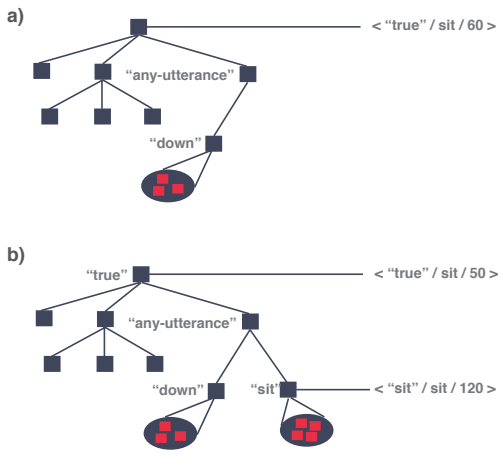

Figure 4: This figure illustrates the process of state-action space discovery. In (a) the trainer begins by rewarding the performance of <"true"/sit>, with the effect being that the reliability and value of <"true"/sit> increases. This in turn increases the frequency of sitting. Once the dog is sitting frequently, the trainer starts saying "sit" as the sit action is performed, while continuing to reward the sit. As the trainer continues this process, the system begins to build a classifier for the specific utterance that occurs during the attention window associated with rewarded sits, and eventually spawns a <"sit-utterance"/sit> percept-action pair (b). Over time the trainer will stop rewarding spontaneous sits or sits in response to other utterances (i.e., <"true"/sit> or <"any-utterance"/sit>). The effect is that the reliability (and value) of these action tuples will drop in comparison with that of <"sit-utterance"/sit> and these less specific and reliable action tuples are expressed less frequently.

During the credit assignment phase, the percept-action pair selected for credit assignment has the option of specializing. Two conditions must be met to be eligible for specialization. First, the value of the percept-action pair must be over some threshold. That is, there needs to be some evidence that the percept-action pair or a variant is potentially valuable. Second, the percept must have a child whose reliability and novelty is above a certain threshold. These statistics essentially provide evidence that a new percept-action pair utilizing that child percept could be more reliable than a percept-action pair relying on the parent percept. If these conditions are met then a new child of the parent percept-action pair is created with the same action as the parent, but with the percept's child. Once added to the parent, it becomes eligible to be selected as the most appropriate representative of all of the percept-action pairs that share its action. The process of specialization is illustrated in Figure 4.

The mechanism described above provides a simple hierarchical search of the state-action space, focusing on those areas that seem most promising and exploring variants of percept-action pairs for which there is evidence that a variant may prove more valuable than its parent.

4.2.3 State Space Discovery

As suggested in Section 3.3, there are important advantages to integrating state space discovery into the learning process. For example, assume a creature is to be taught to perform tricks in response to arbitrary acoustic patterns (utterances, whistles, etc.) If state-space discovery is being performed the only acoustic patterns that need be considered are (a) those that are actually experienced and (b) those for which there is some evidence that they matter with respect to the creature's goals.

An unsupervised technique such as *k-means clustering* can be employed to partition the observed patterns into distinct clusters or classes [Therrien 1989]. In this case, each cluster or class represents a region of the state space. K-means clustering partitions observed patterns into k clusters such that the distance between the center of a cluster and all of the observations that comprise that cluster is minimized across all clusters and patterns. This algorithm is an example of unsupervised learning since the clusters emerge from the data without any supervisory signal providing feedback.

Our experience with dog learning suggests a different approach: treat all patterns that occur contemporaneously with an action that directly leads to a significant outcome (i.e., a reward) as belonging to the same cluster. The action itself becomes the label for the cluster and the reward acts as a natural supervisory signal that indicates if the pattern is a good example either of the cluster in which it was classified (and so should be included in the cluster) or as a seed for a new cluster. This idea is incorporated into the algorithm used in our system, a variation on an incremental k-nearest neighbors technique [Ivanov 2001]. For example, in the case of acoustic processing, there is a percept that recognizes the presence of acoustic patterns, and each of its children percepts represent a cluster of similar patterns. The child percepts are created dynamically as follows:

1. When an acoustic pattern is observed, the acoustic pattern percept and its children responsible for classifying acoustic patterns will attempt to find a match. If a match is found, the associated percept becomes active.

2. If the percept becomes active, the active percept-action pair may change if the percept is referenced by another existing percept-action pair, and if that pair is more reliable in producing good consequences.

3. The pattern is stored in short-term memory.

4. The matching percept's model of the pattern is subsequently updated during credit assignment if:

   (a) The deactivating percept-action pair is directly followed by good consequences.

   (b) The percept is a child of the deactivating percept-action pair's percept and it became active during the percept-action pair's attention window.

   (c) The observation was not classified by one of the percept's children, but 4a) is true. In this case the percept may create a new child and initialize the child's model with the observation as its first sample.

5. Update reliability statistics

For example, assume that initially the acoustic pattern percept has no children, and there is a <"true"/sit> percept-action pair (i.e., "sit") that periodically becomes active. Now suppose that the acoustic pattern percept repeatedly becomes active in the context of a "sit" that consistently leads to a reward. The first time this occurs, it will create a new child percept and initialize it with the pattern that activated it. Every subsequent time that a pattern is detected in the context of a rewarded "sit", that child percept will update its model using the observed pattern. As the child starts classifying incoming patterns correctly (according to its model) within the context of a rewarded "sit", its reliability will increase. Finally, as a result of specialization (Section 4.2.2), when its reliability rises above a threshold, a new percept-action pair will be created, i.e., <"sit"/sit>.

While simple, this algorithm captures what is necessary to learn the kinds of acoustic cues that dogs seem capable of learning. In addition, Ivanov [Ivanov 2001; Ivanov et al. 2001] has explored these ideas more formally and has shown how this simple idea can be incorporated into the well-known Expectation-Maximization learning algorithm as well as SVM. (See [Ivanov et al. 2001] for a detailed discussion of the algorithm used to perform clustering and classification, as well as clustering with a reduced set of examples.)

4.2.4 Action Space Discovery

As suggested in Section 3.3, we can perform action space discovery using almost the same approach as taken for state space discovery. This simplification is made possible by our representation of action (labeled paths through pose space) and by the existence of motor-percepts that can classify a path just taken as being either an example of an existing path or a novel path (Section 4.1.1). Since action space discovery occurs as a result of luring and shaping, however, we need additional machinery. Specifically, luring requires (a) a "follow-your-nose" motor program, (b) a "motor memory" that continuously records recent poses that have been visited and (c) the modification to the credit assignment rule as suggested in Section 3.3. Even though "follow-your-nose" may directly precede a reward, the algorithm can give the credit to another action whose associated path is close to that just taken. Using this idea, the algorithm for performing action space discovery that supports luring is straightforward. When assigning credit (at an action's end):

1. If the creature received a direct reward, compare the path taken to known paths:

   (a) If the path is similar to an existing path, then reward the action associated with that path (i.e., give it the credit) and update the model of the rewarded path using the path just taken as a new example.

   (b) If the path is novel (not well captured by some other action), then create a new motor percept and initialize its model using the path just taken.

2. If no reward is received, ignore the path.

Once a motor-percept is added to the percept tree, reliability statistics are kept just as in the case of other percepts. When a motor-percept's reliability gets above a threshold, a new action tuple is created that uses the motor-percept's path model as its action. Once this is done, the action tuple is a candidate for specialization and can explore to find the context in which it is maximally reliable.

Another kind of motor learning in animals that we have noted is shaping. In our system we adopt a parameterized approach. That is, if the action can be parameterized (e.g., the amplitude of "shake-paw") the parameters can be drawn from a local probability distribution that reflects the pattern of rewards. When an action is about to be performed, a value for the parameter is chosen probabilistically. If the action is subsequently rewarded, the probability distribution is adjusted to make it more likely in the future that a value near the chosen value will be selected. If the action is not rewarded, the probability distribution is either left unchanged or adjusted to make it less likely that a similar value will be chosen in the future.

5 Results and Discussion

The system described in Section 4 has been incorporated into a general-purpose behavior architecture described elsewhere [Burke et al. 2001; Isla et al. 2001]. In the accompanying video figure, we demonstrate some aspects of clicker training and luring on our synthetic pup, as discussed in Section 3 and illustrated in Figures 1 and 5.

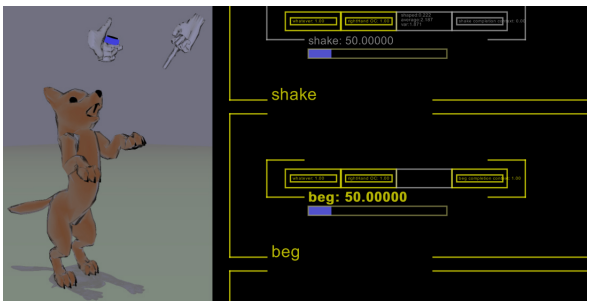

Figure 5: On the left we see Terence performing a Beg. On the right, an example of our action tuple visualizer.

Initially, the pup experiments among its known actions. As the trainer preferentially rewards sitting, the frequency of sitting increases. When sitting is performed reliably, the trainer starts giving the verbal cue "sit" as the pup begins to sit, while also reducing the rate of reinforcement if the pup sits in the absence of the cue. The system, through state space discovery, creates a new percept that contains a model of the (arbitrary) acoustic pattern associated with the rewarded sit and adds it to the pup's percept tree. Eventually, a new percept-action pair is created that represents <"sit"/sit>. At the same time, we see that the frequency of spontaneous sitting decreases.

Next, we demonstrate simple luring of the dog by moving the target hand over the dog's head and clicking as he gets into the sit pose. We also illustrate the more complex example of luring the dog through a novel trajectory—in this case, walking in an 'S' pattern on the ground. When rewarded, this lured trajectory is added to the action space as a new action (through action space discovery), and can thus be associated with a cue and can be selected randomly by the pup in the future just like any of the previously known actions.

Finally, we demonstrate shaping. The pup experiments with different forms of his parameterized "shake-paw" action. The trainer rewards ever higher versions of the shake action until the pup shakes his paw high reliably.

5.1 Limitations and Future Work

Our system has a number of important limitations and areas for future work:

- The system is biased to learn immediate consequences rather than extended sequences. Nonetheless, learning sequences is important, and we will be addressing this area in our future work.
- The system does not address spatial and social learning. Our sense is that while much can be shared across learning tasks, it is very likely that the right solution will have specialized mechanisms and representations for specific learning tasks. (See [Isla 2001] for an example of spatial learning.)
- There are things the system should be able to learn which it cannot—for example, states that are conjunctions or disjunctions of percepts. In addition, it cannot generalize from specific percepts to more general ones. These, however, are hard problems. An easier problem, and one that has been addressed by a variant of the system discussed here, is to learn important correlations among events that enable the creature to act proactively[Burke 2001].
- The existence, speed and quality of classifiers, such as our utterance or path classifiers, are critically important to the functioning of the system, but we have only touched on them briefly here. While our integrated approach helps the classifiers build better models, more could be done. For example, the classifiers do not currently make use of negative examples. (See [Ivanov 2001] for an in-depth discussion of this topic.)
- How will the system scale? We feel that our integrated approach as well as our hierarchical representations of the learning spaces will allow our system to scale better than a traditional RL system, but more work needs to be done to support this claim.

5.2 Useful Insights

While our results are from a specific learning system, there are a number of ideas that we believe are generally useful in the context of learning for synthetic characters, regardless of the specifics of the implementation.

- **Use temporal proximity to limit search.** We utilize a temporal attention window that overlaps the beginning of an action to identify potentially relevant states. Similarly, we generally assign credit to the action that immediately precedes a motivationally significant event.
- **Use hierarchical representations of state, action and state-action space.** We utilize loosely hierarchical representations of state, action and state-action space and use simple statistics to identify potentially promising areas of the respective spaces for exploration. We grow these hierarchies downward toward more fine-grained representations of state and more specific (and hopefully more reliable) state-action pairs.
- **Use natural feedback signals to guide exploration of the three spaces**. The practical effect in both cases is that fewer models are built, and those that are built tend to be more relevant and robust.
- **Bias frequency and variability of action so as to facilitate learning.** This not only allows the creature to exploit what it knows, but also gives it more opportunities to discover more reliable variations.
- **Give credit where credit is due**. The state-action pair that would normally receive credit should be given the option to delegate its credit to another, potentially more appropriate, state-action pair. We saw that this was particularly useful in the context of "luring".

6 Conclusion

We described a practical approach to real-time learning for synthetic characters that allows them to learn the same kinds of things that dogs seem to learn so easily. We believe that by embedding dog-level learning into synthetic characters, we can provide them with a way to meaningfully adapt to human interaction.

By addressing the three problems of state, action, and state-action space discovery at the same time, the solution for each becomes easier. Similarly, by viewing learning and training as a coupled system we were able to gain valuable insights into each.

7 Acknowledgments

The authors wish to thank the other members and friends of the Synthetic Characters Group, past and present who contributed to this work, including: Robert Burke, Damian Isla, Geoff Beatty, Jennie Cochran, Scott Eaton, Ashley Eden, Jesse Gray, Matthew Grimes, Michal Hlavac, Chris Kline, Derek Lyons, Ben Resner, Ken Russell, Adolph Wong, and Soon-Yee Yoon. Thanks also to Prof. Gerald Schneider. We are indebted to Gary Wilkes for his insights into dog training and behavior. We would especially like to thank Sydney, the first author's silky terrier, who continues to be the best teacher of all. This work was funded by the Digital Life Consortium of the MIT Media Lab.

References

BALLARD, D. 1997. *An Introduction to Natural Computation*. MIT Press, Cambridge, MA.

BLUMBERG, B., AND GAYLEAN, T. 1995. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of SIGGRAPH 1995*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM.

BURKE, R., ISLA, D., DOWNIE, M., IVANOV, Y., AND BLUMBERG, B. 2001. Creature smarts: The art and architecture of a virtual brain. In *Proceedings of the Computer Game Developers Conference*.

BURKE, R. 2001. *Its about Time:Temporal Representation for Synthetic Characters*. Master's thesis, The Media Lab, MIT.

COPPINGER, R., AND COPPINGER, L. 2001. *Dogs: A Startling New Understanding of Canine Origin, Behavior, and Evolution*. Scribner, New York, NY.

DOWNIE, M. 2000. *behavior, animation, music: the music and movement of synthetic characters*. Master's thesis, The Media Lab, MIT.

DRESCHER, G. 1991. *Made-Up Minds:A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge MA.

EVANS, R. 2002. Varieties of learning. In *AI Game Programming Wisdom*, E. Rabin, Ed. Charles River Media, Hingham MA.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOLOUS, D. 2001. Composible controllers for physics-based character animation. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM.

FUNGE, J., TU, X., AND TERZOPOLOUS, D. 1999. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In *Proceedings of SIGGRAPH 1999*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM.

GALLISTEL, C. R., AND GIBBON, J. 2000. Time, rate and conditioning. *Psychological Review 107*.

GLEICHER, M. 1998. Retargetting motion to new characters. In *Proceedings of SIGGRAPH 1998*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM.

GOULD, J., AND GOULD, C. 1999. *The Animal Mind*. W.H. Freeman, New York, NY.

GRAND, S., CLIFF, D., AND MALHOTRA, A. 1996. Creatures: Artificial life autonomous agents for home entertainment. In *Proceedings of the Autonomous Agents '97 Conference*.

GRZESZCZUK, R., AND TERZOPOULOS, D. 1995. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of SIGGRAPH 1995*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM.

GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. 1998. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of SIGGRAPH 1998*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM.

HODGINS, J., AND POLLARD, N. 1997. Adapting simulated behaviors for new characters. In *Proceedings of SIGGRAPH 1997*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM.

ISLA, D., BURKE, R., DOWNIE, M., AND BLUMBERG, B. 2001. A layered brain architecture for synthetic creatures. In *Proceedings of The International Joint Conference on Artificial Intelligence*.

ISLA, D. 2001. *The Virtual Hippocampus: Spatial Common Sense for Synthetic Creatures.* Master's thesis, The Media Lab, MIT.

IVANOV, Y., BLUMBERG, B., AND PENTLAND, A. 2001. Expectation maximization for weakly labeled data. In *Proceedings of the 18th International Conference on Machine Learning*.

IVANOV, Y. 2001. *State Discovery for Autonomous Creatures*. PhD thesis, The Media Lab, MIT.

KAELBLING, L. 1990. *Learning in embedded systems*. PhD thesis, Stanford University.

KAPLAN, F., OUDEYER, P.-Y., KUBINYI, E., AND MIKLOSI, A. 2001. Taming robots with clicker training : a solution for teaching complex behaviors. In *Proceedings of the 9th European workshop on learning robots, LNAI*, Springer, M. Quoy, P. Gaussier, and J. L. Wyatt, Eds.

LINDSAY, S. 2000. *Applied Dog Behavior and Training*. Iowa State University Press, Ames, IA.

LORENZ, K., AND LEYAHUSEN, P. 1973. *Motivation of Human and Animal Behavior: An Ethological View*. Van Nostrand Reinhold Co., New York, NY.

LORENZ, K. 1981. *The Foundations of Ethology*. Springer-Verlag, New York, NY.

MITCHELL, K. 1997. *Machine Learning*. McGraw Hill, New York, NY.

PERLIN, K., AND GOLDBERG, A. 1996. Improv: A system for scripting interactive actors in virtual worlds. In *Proceedings of SIGGRAPH 1996*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM.

PRYOR, K. 1999. *Clicker Training for Dogs*. Sunshine Books, Inc., Waltham, MA.

RABINER, L., AND JUANG, B.-H. 1993. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ.

RAMIREZ, K. 1999. *Animal Training:Successful Animal Management Through Positive Reinforcement*. Shedd Aquarium, Chicago, IL.

RESNER, B., STERN, A., AND FRANK, A. 1997. The truth about catz and dogz. In *The Computer Games Developer Conference*.

REYNOLDS, C. 1987. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of SIGGRAPH 1987*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM.

ROSE, C., COHEN, M., AND BODENHEIMER, B. 1999. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics And Applications 18*, 5.

SHETTLEWORTH, S. J. 1998. *Cognition, Evolution and Behavior*. Oxford University Press, New York, NY.

SUTTON, R., AND BARTO, A. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge MA.

SUTTON, R. 1991. Reinforcement learning architectures for animats. In *The First International Conference on Simulation of Adaptive Behavior*, MIT Press, Paris, Fr.

THERRIEN, C. 1989. *Decision Estimation and Classification: An Introduction to Pattern Recognition and Related Topics*. John Wiley and Sons, New York, NY.

TOMLINSON, B., AND BLUMBERG, B. 2002. Alphawolf: Social learning, emotion and development in autonomous virtual agents. In *First GSFC/JPL Workshop on Radical Agent Concepts*.

TU, X., AND TERZOPOULOS, D. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH 1994*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM.

VAN DE PANNE, M., AND FIUME., E. 1993. Sensor-actuator networks. In *Proceedings of SIGGRAPH 1993*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM.

VAN DE PANNE, M., KIM, R., AND FIUME., E. 1994. Synthesizing parameterized motions. In *5th Eurographics Workshop on Simulation and Animation*.

WATKINS, C. J., AND DAYAN, P. 1992. Q-learning. *Machine Learning 8*.

WILKES, G. 1995. *Click and Treat Training Kit*. Click and Treat Inc., Mesa, AZ.

YOON, S., BLUMBERG, B., AND SCHNEIDER, G. 2000. Motivation-driven learning for interactive synthetic characters. In *Proceedings of the Fourth International Conference on Autonomous Agents*.

YOON, S., BURKE, R., AND BLUMBERG, B. 2000. Interactive training for synthetic characters. In *Proceedings of AAAI 2000*.