Lifelong Robot Learning[1]

Sebastian Thrun[2] and Tom M. Mitchell[3]

[2] University of Bonn, Institut für Informatik III, Römerstr. 164, 53117 Bonn, Germany
[3] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract. Learning provides a useful tool for the automatic design of autonomous robots. Recent research on learning robot control has predominantly focussed on learning single tasks that were studied in isolation. If robots encounter a multitude of control learning tasks over their entire lifetime, however, there is an opportunity to transfer knowledge between them. In order to do so, robots may learn the invariants of the individual tasks and environments. This task-independent knowledge can be employed to bias generalization when learning control, which reduces the need for real-world experimentation. We argue that knowledge transfer is essential if robots are to learn control with moderate learning times in complex scenarios. Two approaches to lifelong robot learning which both capture invariant knowledge about the robot and its environments are presented. Both approaches have been evaluated using a HERO-2000 mobile robot. Learning tasks included navigation in unknown indoor environments and a simple find-and-fetch task.

1 Why Learning?

Traditionally, it has often been assumed in robotics research that accurate a priori knowledge about the robot, its sensors, and most important its environment is available. By assuming the availability of accurate models of both the world and the robot itself, the kind of environments such robots can operate in, and consequently the kind of tasks such robots can solve, is limited. Limitations especially arise from four factors:

- **Knowledge bottleneck.** A human designer had to provide accurate models of the world and the robot.
- **Engineering bottleneck.** Even if sufficiently detailed knowledge is available, making it computer-accessible,i.e., hand-coding explicit models of robot hardware, sensors and environments, has often been found to require unreasonable amounts of programming time.
- **Tractability bottleneck.** It was early recognized that many realistic robot domains are too complex to be handled efficiently [Schwartz *et al.*, 1987], [Canny,

[1] This paper is also available as Technical Report IAI-TR-93-7, University of Bonn, Dept. of Computer Science III, March 1993.

1987], [Bylander, 1991]. Computational tractability turned out to be a severe obstacle for designing control structures for complex robots in complex domains, and robots were far from being "reactive."

- **Precision bottleneck.** The robot device must be precise enough to accurately execute plans that were generated using the internal models of the world.

Recent research on autonomous robots has changed the design of autonomous agents and pointed out a promising direction for future research in robotics (see for example [Brooks, 1989] and several papers in [Maes, 1991]). Reactivity and real-time operation have received considerably more attention than, for example, optimality and completeness. Many approaches have dropped the assumption that perfect world knowledge is available–some systems even operate in the extreme where no domain-specific initial knowledge is available at all. Consequently, today's robots are facing gradually unknown, hostile environments, they have to orient themselves, explore their environments autonomously, recover from failures, and they have to solve whole families of tasks.

If robots lack initial knowledge about themselves and their environments, learning becomes inevitable. The term *learning* refers to a variety of algorithms that are characterized by their ability to replace missing or incorrect world knowledge by experimentation, observation, and generalization. Learning robots thus collect parts of their domain knowledge themselves and improve over time. They are less dependent on a human instructor to provide this knowledge beforehand. A learning robot architecture is typically flexible enough to deal with a whole class of environments, robots and/or tasks (goals). Consequently the internal prior knowledge, if available at all, is often too weak to solve a concrete problem off-line. In order to reach a goal, learning robots rely on the interaction with their environment to extract information. Different learning strategies differ mainly in three aspects: their way to do experimentation (exploration), their way to generalize from the observed experience, and the type and amount of prior knowledge that constrains the space of their internal hypothesizes about the world. There will be no optimal, general learning technique for autonomous robots, since learning techniques are characterized by a trade-off between the degree of flexibility, given by the size of the "gaps" in the domain knowledge, and the amount of observations required for filling these gaps. Generally speaking, the more universal a robot learning architecture, the more experimentation we expect the robot to take to learn successfully, and vice versa. However, the advantage of applying learning strategies to autonomous robot agents is obvious: Learning robots can operate in a whole class of initially unknown environments, and they can compensate for changes, since they can re-adjust their internal belief about the environment and themselves. Moreover, all empirically learned knowledge is grounded in the real world. It has long been recognized in AI that learning is a key feature for making autonomous agents capable of solving more complex tasks in more realistic environments. Recent research has produced a variety of rigorous learning techniques that allow a robot to acquire huge chunks of knowledge by itself. See for example [Mel, 1989], [Moore, 1990], [Pomerleau, 1989], [Tan, 1991],

Figure 1 The robot control learning problem. A robot agent is able to perceive the state of its environment by its sensors, and change it using its effectors (actions). A reward function that maps sensations to rewards measures the performance of the robot. The control learning problem is the problem of finding a control policy that generates actions such that the reward is maximized over time.

[Mahadevan and Connell, 1991], [Lin, 1992b], and [Kuipers and Byun, 1990].

What exactly is the concrete problem addressed by robot learning? Let us outline a general definition of learning robot control. Assume the robot acts in an environment W (the *world*). Each time step, the environment is in a certain state $z \in Z$. By state we mean the sum of all quantities in the environment that may change during the lifetime of the robot. The robot is able to (partially) perceive the state of its environment through its sensors. It is also able to act using its effectors, as shown in Figure 1. Let S defines the set of all possible sensations, and A the set of all actions that the robot can execute. Actions (including zero-actions, if the robot does nothing) change the state of the world. Hence, the environment can be understood as a mapping $W : Z \times A \longrightarrow Z$ from states and actions to states. For example, imagine an autonomous mobile robot whose task it is to keep the floor clean. The worlds such a robot will be acting in are buildings, including the obstacles surrounding the robot, the floors, the dirt on the floors, humans that walk by, and so on. Appropriate sensors might include a camera mounted on the robot's arm, and the set of all sensations might be the space of all camera images. Actions could be "go forward," "turn," "switch on/off vacuum," and "lift arm."

In order to define the goals of the robot, we assume that the robot is given a *reward function* $R : S \longrightarrow I\!R$, that maps sensations to scalar reward values. The reward evaluates the success of the robot to solve its tasks: In the most simple form the reward is positive (say 100), if the robot reaches its goals, it is negative (-100) if the robot fails, and zero otherwise. Positive reward corresponds to pleasure, and negative reward represents pain. In the mobile robot domain, for example, the reward may be positive if there is no dirt on the floor, and negative reward may be received if the robot collides with the furniture or runs out of battery power. The control learning problem is the problem of finding a control function F that generates actions such that the reward is maximized over time. More formally, a *control learning problem* can be described in the following way:

Control learning problem:
$\langle S, A, W, R \rangle \longrightarrow F : S^* \rightarrow A$
such that F maximizes R over time

This formulation of the robot control learning problem has been extensively studied in the field of "Reinforcement Learning." [Sutton, 1984], [Barto *et al.*, 1991]. Thus far, most approaches that emerged from this field have studied robot learning with a minimal set of assumptions: The robot is able to sense, it is able to execute actions, actions have an effect on future sensations, and there is a pre-given reward function that defines the goals of the robot. The goal of reinforcement learning is to maximize the reward over time. Note that this general definition lacks any specifications about the robot at hand, its environment, or the kind of reward functions the robot is expected to face. Hence, approaches that are able to adaptively solve such a robot control learning problem are *general* learning techniques. Perhaps the most restricting assumption found in most approaches to reinforcement learning is that the robot is able to sense the state of the world reliably. If this is the case, it suffices to learn the policy as a function of the most recent sensation to action: $F : S \rightarrow A$, i.e., the control policy is purely reactive. As Barto et al. pointed out [Barto *et al.*, 1991], the problem of learning a control policy can then be attacked by Dynamic Programming techniques [Bellman, 1957].

As it turns out, even if robots have access to complete state descriptions of their environments, learning control in complex robot worlds with large state spaces is practically not feasible. This is because it takes often too much experimentation to acquire the knowledge required for maximizing reward, i.e., to fill the huge knowledge gaps, and robot hardware is slow. One might argue that better learning techniques have to be invented that decrease the learning complexity, while still being general. Although there is certainly space for better learning and generalization techniques that gradually decrease the amount of experimentation required, it seems unlikely that such techniques will ever be applicable to complex, real-world robot environments with sparse reward and difficult tasks. The complexity of knowledge acquisition is inherent in the formulation of the problem. Real-world experimentation will be the central bottleneck of any general learning technique that does not utilize prior knowledge about the robot and its environment.

Why do natural agents such as humans learn so much better than artificial agents? Maybe a better question to ask is: Is the learning problem faced by natural agents any simpler than that of artificial ones? We will not attempt to give general answers to these general questions. We will, however, point out the importance of knowledge transfer and lifelong learning problems in order to make robots to learn more complex control.

2 The Necessity of Lifelong Agent Learning

Humans typically encounter a multitude of control learning problems over their entire lifetime, and so do robots. Henceforth, in this paper we will be interested in the lifelong learning problem faced by a robot agent, in which it must learn a collection of control policies for a variety of related control tasks. Each of these control problems, $\langle S, A, W_i, R_i\rangle$ involves the same robot with the same set of sensors, effectors, and may vary only in the particular environment W_i, and in the reward function R_i that defines the goal states for this problem. For example, an industrial mobile robot might face multiple learning tasks such as shipping packages, delivering mail, supervising critical processes, guarding its work-place at night, and so on. In this scenario the environment will be the same for all tasks, but the reward function varies. Alternatively, a window-cleaning robot that is able to climb fronts of buildings and move arbitrarily on walls and windows might have the very single task of cleaning windows. It will, however, face multiple fronts and windows (environments) over its lifetime.

In the lifelong learning problem, for each different environment and each reward function the robot agent must find a different control policy, F_i. The lifelong learning problem of the agent therefore corresponds to a set of control learning problems:

> *Lifelong learning problem:*
> $\{\langle S, A, W_i, R_i\rangle \;\longrightarrow\; F_i | F_i : S \rightarrow A\}$
> *such that F_i maximizes R_i over time*

Of course the agent could approach the lifelong learning problem by handling each control learning problem independently. However, there is an opportunity for the agent to do considerably better. Because these control learning problems are defined in terms of the same S, A, and potentially the same W, the agent should be able to reduce the difficulty of solving the i-th control learning problem by using knowledge it acquired from solving earlier control learning problems. For example, a robot that must learn to deliver laser printer output and to collect trash in the same environment should be able to use much of what it has learned from the first task to simplify learning the second. The problem of lifelong learning offers the opportunity for synergy among the different control learning problems, which can speed-up learning over the lifetime of the agent.

Viewing robot learning as a lifelong learning problem motivates research on "bootstrapping" learning algorithms that transfer learned knowledge from one learning task to another. Bootstrapping learning algorithms might start with low-complexity learning problems, and gradually increase the problem solving power to harder problems. Similar to humans, future robots might first have to learn simple tasks (such as low-level navigation, hand-eye coordination), and, once successful, draw their attention to increasingly more difficult and complex learning tasks (such as picking up

Figure 2 The robot we used in all experiments described in this paper is a wheeled HERO-2000 robot with a manipulator and a gripper. It is equipped with two sonar sensors, one on the top of the robot that can be directed by a rotating mirror to give a full 360° sweep (24 values), and one on the wrist that can be rotated by the manipulator to give a 90° sweep. Sonar sensors return approximate echo distances. Such sensors are inexpensive but very noisy.

---

and delivering laser printer output). As, for example, Singh [Singh, 1992b] and Lin [Lin, 1992b] demonstrate, learning related control tasks with increasing complexity can result in a remarkable synergy between these control learning tasks, and an improved problem solving power. In their experiments, simulated mobile robots were able to solve more complex navigation tasks when the robots were given simpler, related learning tasks beforehand. They also report that their systems were unable to learn the same complex tasks in isolation, pointing out the importance of knowledge transfer for robot learning.

The reminder of the paper is organized as follows. In the next two sections, we will briefly present two approaches to the lifelong learning problem. Both approaches have been implemented and evaluated using a HERO-2000 mobile robot with a manipulator shown in Figure 2. In the first approach, called explanation-based neural network learning (EBNN), we will assume that the environment of the agent stays the same for all control learning tasks. This allows the robot to learn task-independent action models. Once learned, these action models provide a means of transferring knowledge across control learning tasks. In EBNN, such action models are used to *explain and analyze* observations. In Section 4, we drop the assumption of static environments. Instead, we describe a mobile autonomous robot that has to solve the

Figure 3 Episode: Starting with the initial state s_1, the action sequence $a_1, a_2, \ldots, a_{n-1}$ was observed to produce the final reward R_n. The robot agent uses its action models, which capture previous experiences in the same environment, to *explain* the observed episode and thus bias learning. See text for an explanation.

---

same task in different, related environments. We demonstrate how this robot might learn and transfer environment-independent knowledge that captures the characteristics of its sensors, as well as invariant characteristics of the environments. In Section 5 we will review some related approaches to robot learning which also utilize previously learned knowledge. As we will see, there are several types of techniques that can be grouped into categories. The paper is concluded by Section 6

3 Explaining Observations in Terms of Prior Experiences

As defined in the previous section, the lifelong learning problem faced by a learning robot agent is the problem of learning collections of tasks in families of environments over the entire lifetime. In this section we will draw our attention to a particular subtype of lifelong learning problems, namely to the lifelong learning problem of robots that spend their whole life exclusively in the same environment. This restricted class of lifelong learning scenarios plays an important role in autonomous agent research. Many prospective robot agents, such as housekeeping robots, industrial robot arms or artificial insects, face a variety of learning problems in the very same environment. In such scenarios, knowledge about the environment can be reused, since the environment stays the same for each task. The explanation-based neural network learning algorithm (EBNN), which will be presented in this section, transfers task-independent knowledge via learned action models which are learned empirically during problem solving.

3.1 Learning Action Models

Robots observe their environments (and themselves) by the effects of their actions. In other words, each time an action is performed, the robot may use its sensors to sense the way the world has changed. Let us assume for simplicity that the robot is able to accurately sense the state of the world. As pointed out in the previous section, learning control reduces to learning a reactive controller[4]. If the environment is sufficiently predictable, neural network learning techniques such as the Back-propagation training procedure [Rumelhart *et al.*, 1986] can be used to model the environment. More specifically, each time an action is performed, the previous state denoted by s, the action a and the next state s' form a training example for the action model network, denoted by M:

$$\langle\, s, a \,\rangle \longrightarrow s'$$

Action models[5] are thus functions of the type $M : S \times A \longrightarrow S$. Since we assume that the environment is the same for all tasks, action models capture invariants that allow for transferring knowledge from one task to another.

Each individual control learning problem requires a different policy, i.e., learning a control function $F_i : S \longrightarrow A$ that, when employed by the agent, maximizes the corresponding reward R_i. In general, F_i is difficult to learn directly. Following the ideas of reinforcement learning ([Samuel, 1959], [Sutton, 1988], [Barto *et al.*, 1990], [Watkins, 1989]), we decompose the problem of learning F_i into the problem of learning an *evaluation function*, $\mathcal{Q}_i$, defined over states and actions.

> $\mathcal{Q}_i : S \times A \longrightarrow I\!R$, where $\mathcal{Q}_i(s, a)$ is the expected future cumulative reward achieved after executing action a in state s.

Assume for a moment the agent had learned an accurate evaluation function $\mathcal{Q}_i$. Then it can easily use this function to select optimal actions that maximize reward over time. Given some state, s, that the agent finds itself in, it computes its control action a^* by considering its available actions to determine which of them produces the highest $\mathcal{Q}_i$ value:

$$a^* = \text{argmax}_{a \in A} \;\; \mathcal{Q}_i(s, a)$$

Since $\mathcal{Q}_i(s, a)$ measures the future cumulative reward, a^* is the optimal action if $\mathcal{Q}_i(s, a)$ is correct. The problem of learning a policy is thus reduced to the problem of learning an evaluation function.

How can an agent *learn* the evaluation function $\mathcal{Q}_i$? To see how training data for learning $\mathcal{Q}_i$ might be obtained, consider the scenario depicted in Figure 3. The robot

---

[4] We intentionally avoid the complex problem of incomplete and noisy perception perception, since the algorithm presented in this section is kind of orthogonal to research on these issues. See [Bachrach and Mozer, 1991], [Chrisman, 1992], [Lin and Mitchell, 1992], [Mozer and Bachrach, 1989], [Rivest and Schapire, 1987], [Tan, 1991], [Whitehead and Ballard, 1991] for approaches to learning with incomplete perception.

[5] See for example [Barto *et al.*, 1989], [Jordan, 1989], [Munro, 1987], [Thrun, 1992] for more approaches to learning action models with neural networks.

agent begins at the initial state s_1 and performs the action sequence $a_1, a_2, \ldots, a_n$. After action a_n it receives the reward $r_n = 92.3$ which, in this example, indicates that the action sequence was considerably successful. At a first glance, this episode can be used by the agent to derive training examples for the evaluation function $\mathcal{Q}_i$ by associating the final reward[6] with each state-action pair:

$$\begin{aligned}
\langle\ s_1, a_1\ \rangle &\longrightarrow r_n\ = 92.3 \\
\langle\ s_2, a_2\ \rangle &\longrightarrow r_n\ = 92.3 \\
&\ \ \vdots \\
\langle\ s_{n-1}, a_{n-1}\ \rangle &\longrightarrow r_n\ = 92.3
\end{aligned} \tag{1}$$

An inductive learning method, such as the Back-propagation training procedure, can use such training examples to learn $\mathcal{Q}_i$. As the number of training examples grows, the agent's internal version of $\mathcal{Q}_i$ will improve, resulting in it choosing increasingly effective actions.[7] Notice that for each control learning problem $\langle S, A, W_i, R_i\rangle$, the agent must learn a distinct $\mathcal{Q}_i$, since the reward differs for different tasks.

3.2 The Explanation-Based Neural Network Learning Algorithm

How can the agent use its previously learned knowledge, namely the neural network action models, to guide learning of the evaluation function $\mathcal{Q}_i$? As pointed out earlier in this paper, we are interested in the lifelong learning problem. Since we assume throughout this section that the environment is the same for all individual control learning problems, neural network action models capture important domain knowledge that is independent of the particular control learning problem at hand. In the explanation-based neural network learning algorithm (EBNN), the agent uses these action models to bias learning of the control functions.

[6] For simplification of the notation, we assume that reward will only be received at the end of an episode. EBNN can be applied to arbitrary reward functions. See [Mitchell and Thrun, 1993b] for more details.

[7] Note that more sophisticated learning schemes for learning evaluation functions have been developed. In his dissertation, Watkins [Watkins, 1989] describes $\mathcal{Q}$-Learning, a scheme for learning evaluation function $\mathcal{Q}_i(s_k, a_k)$ *recursively*. In $\mathcal{Q}$-Learning training patterns are derived based on the maximum possible $\mathcal{Q}$ value at the next state: $\langle s_k, a_k\rangle \longrightarrow \max_a \mathcal{Q}(s_{k+1}, a)$. Indeed, in all our experiments we applied a linear combination of Watkins' recursive scheme and the non-recursive scheme described in the paper. This combination is strongly related to Sutton's $TD(\lambda)$ algorithms [Sutton, 1988]. Since the exact procedure is not essential for the ideas presented in this paper, we will omit any details. A second extension, also used widely, is to discount reward over time. If actions are to be chosen such that the number of actions is minimal, reward is typically discounted with a *discount factor* $\gamma \leq 1$. The resulting control policy consequently prefers sooner reward to more distant reward. See [Mitchell and Thrun, 1993b] or [Thrun and Mitchell, 1993] for a more detailed description of these issues.

Figure 4 Fitting slopes: Let f be a target function for which three examples $\langle x_1, f(x_1)\rangle$, $\langle x_2, f(x_2)\rangle$, and $\langle x_3, f(x_3)\rangle$ are known. Based on these points the learner might generate the hypothesis g. If the output-input derivatives are also known, the learner can do much better: h.

EBNN works as follows. Suppose the robot faces the control learning problem number i, i.e., it has to learn the evaluation function Q_i. The learning scheme described above provides training values for the desired evaluation function. Repetitive experimentation allows the robot to collect enough data to learn the desired Q_i. However, this process does not utilize the knowledge represented in the action models. Assume the agent has learned already accurate action models that model the effect of actions on the state of the environment. Of course, these action models will only be approximately correct, since they are learned inductively from a finite amount of training data. In EBNN, the agent employs these action models to explain, analyze and generalize better from the observed episodes. This is done in the following three steps:

1. **Explain.** An *explanation* is a post-facto prediction of the observed state-action sequence [DeJong and Mooney, 1986], [Mitchell *et al.*, 1986], [Mitchell and Thrun, 1993a]. Starting with the initial state-action pair (s_1, a_1), the agent post-facto predicts subsequent states up to the final state s_n using its neural network action models. Since the action models are only approximately correct, predictions will deviate from the observed states.
2. **Analyze.** Having explained the whole episode, the explanation is *analyzed* to extract further information that is useful for learning the evaluation function Q_i. In particular, the agent analyzes how a small change of the states features will affect the final reward, and thus the value of the evaluation function. This is done by extracting *partial derivatives* (*slopes*) of the target function Q_i with respect to the observed states in the episode: First, the agent computes the partial derivative of the final reward with respect to the final state s_n. Notice that the reward function $R(s)$, including its derivative, is assumed to be given to the agent. These slopes are now propagated backwards through the chain of action model inferences. Neural network action models represent differentiable functions. Using the chain rule of differentiation, the agent computes the partial derivative of the final reward with respect to the preceding state s_{n-1} by multiplying the partial derivative of the reward with the derivative of the neural network action

model. This process is iterated, yielding all partial derivatives of the final reward along the whole episode:

$$\frac{\partial Q_i}{\partial s_n}(s_{n-1}, a_{n-1}) \approx \frac{\partial R}{\partial s}(s_n) \cdot \frac{\partial M}{\partial s}(s_{n-1}, a_{n-1})$$

$$\vdots$$

$$\frac{\partial Q_i}{\partial s_1}(s_1, a_1) \approx \frac{\partial R}{\partial s}(s_n) \cdot \frac{\partial M}{\partial s}(s_{n-1}, a_{n-1}) \cdot \cdots \cdot \frac{\partial M}{\partial s}(s_2, a_2) \cdot \frac{\partial M}{\partial s}(s_1, a_1)$$

Here $M : S \times A \longrightarrow S$ denotes the neural network action model. The reward-state slopes analyze the importance of the state features for the final reward. State features believed (by the action models) to be irrelevant to achieving the final reward will have partial derivatives of zero, whereas large derivative values indicate the presence of strongly relevant features.

3. **Learn.** The analytically extracted slopes approximate the slopes of the target evaluation function Q_i. Figure 4 illustrates the importance of the slope information of the target function. Suppose the unknown target function is the function f depicted in Figure 4a, and suppose that three training examples are given: x_1, x_2 and x_3. An arbitrary continuous function approximator, for example a neural network, might hypothesize the function g shown in Figure 4b. If the slopes at these points are known as well, then the resulting function might be much better, as illustrated in Figure 4c.
EBNN uses a combined learning scheme utilizing both types of training information. The target values (c.f. Equation 2) for learning Q_i are generated from observation, whereas the target slopes, given by

$$\nabla \langle s_{n-1}, a_{n-1} \rangle \longrightarrow \frac{\partial R}{\partial s}(s_n) \cdot \frac{\partial M}{\partial s}(s_{n-1}, a_{n-1})$$

$$\vdots$$

$$\nabla \langle s_1, a_1 \rangle \longrightarrow \frac{\partial R}{\partial s}(s_n) \cdot \prod_{k=1}^{n-1} \frac{\partial M}{\partial s}(s_k, a_k),$$

are extracted from analyzing the observations using domain knowledge acquired in previous control learning tasks. Both sources of training information, the target values and the target slopes, are used to update the weights and biases of the target network.[8] Consequently, the domain knowledge is used to bias the generalization. Since this bias is knowledgeable, it will partially replace the need for real-world experimentation, hence accelerate learning.

[8] As Simard and colleagues pointed out, the Back-propagation algorithm can be extended to fit target slopes as well as target values [Simard *et al.*, 1992]. Their algorithm *tangent prop* incrementally updates weights and biases of a neural network such that both the value and the slope error are simultaneously minimized.

3.3 Accommodating Imperfect Action Models

Initial experiments with EBNN on a simulated robot navigation task showed a significant speedup in learning when the robot agent had access to highly accurate action models [Mitchell and Thrun, 1993b]. If the action models are not sufficiently accurate, however, the robot performance can seriously suffer from the analysis. This is because the extracted slopes might very well be wrong and mislead generalization. This observations raises an essential question for research on lifelong agent learning and knowledge transfer: How can a robot agent deal with incorrect prior knowledge? Clearly, if the agent lacks training experience, any inductively learned bias might be poor and misleading. But even in the worst case, a learning mechanism that employs previously learned domain knowledge should not take more time for learning control than a learning mechanism that does not utilize prior knowledge at all. How can the learner avoid the damaging effects arising from poor prior knowledge?

In EBNN, malicious slopes are identified and their influence is gradually reduced. More specifically, the accuracy of the extracted slopes is estimated based upon the observed prediction error of the action models. For example, if the action models perfectly post-facto predict the observed episode, the estimated accuracy of all slopes will be 1. Likewise, if for some of the action models in the chain of model derivatives have produced inaccurate state predictions, the corresponding estimated accuracy will be close to zero. The accuracies of the slopes are now used when training the target network. Since tangent prop allows to weight each training pattern individually, the estimated accuracies can be used to determine the ratio with which value learning and slope learning are weighted when learning the target concept. More specifically, in EBNN the step-size for weight updates is multiplied by the estimated slope accuracy when learning slopes.

As illustrated elsewhere [Mitchell and Thrun, 1993b], weighting slope training by their accuracies was found to successfully reduce the impact of malicious slopes resulting from inaccurate action models. We evaluated EBNN using nine different sets of action models that were trained with different amounts of training data. With well-trained action models in the simulated robot domain, the same speedup was observed as before. With increasing inaccurate action models, the performance of EBNN approached that of standard reinforcement learning without knowledge transfer. In these experiments, EBNN degraded gracefully with increasing errors in the action models. These results are intriguing since they indicate the feasibility of lifelong learning algorithms that are able to benefit from previously learned bias, even if this bias is poor.

3.4 A Concrete Example: Learning to Pick up a Cup

We will present some initial results obtained with our HERO-2000 robot. Thus far, we predominantly investigated the effect of previously learned knowledge on the

Figure 5 The robot uses its manipulator and a sonar sensor to sense the distance to a cup and to pick it up.

---

learning speed for new control learning tasks. We therefore trained the action models beforehand with manually collected training data. The robot agent had to learn a policy for approaching and grasping a cup. The robot at hand, shown again in Figure 5, used a hand-mounted sonar sensor to observe its environment. Sonar data was preprocessed to estimate the direction and distance to the closest object, which was used as the world state description. Robot actions were `forward(inches)`, `turn(degrees)`, and `grab`. Positive reward was received for successful grasps, and negative reward for unsuccessful grasps as well as for losing sight of the cup. In this experiment, actions were modeled by three separate networks, one for each action. The networks for the parameterized actions `forward(inches)` and `turn(degrees)` predicted the distance and the orientation to the cup (one hidden layer with 6 units), whereas the model for `grab` predicted the probability that a pre-given, open-loop grasping routing would manage to pick up the cup (four hidden units). All action models were pre-learned from approximately two hundred training episodes containing an average of five steps each, which were manually collected beforehand. Figure 6 shows as an example the action model for the `grab` action. Since this particular action model modeled the probability of success of the grasping routine, the reward function was simply the identity mapping $R(s) = s$ (with the constant derivative 1). The evaluation function Q was also modeled by three distinct networks, one for each action (8 hidden units each). After learning the action models, the six training episodes for the evaluation networks shown in Figure 7 were provided by a human teacher who controlled the robot. We applied a version of $TD(\lambda)$ [Sutton, 1988] with $\lambda = 0.7$ and Watkins' Q-Learning [Watkins, 1989]

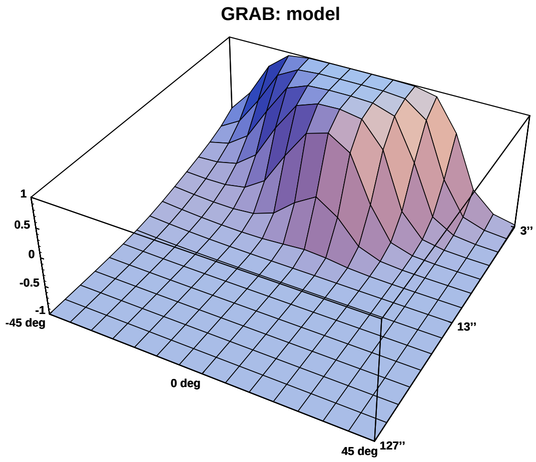

Figure 6 Action model for the action model `grab`. The x and y axis measures again the angle and distance to the cup. The z axis plots the expected success of the `grasp` action, i.e., the probability that the grasping succeeds.

---

with experience replay [Lin, 1992a] for learning control.

Figure 8 illustrates the learned Q function for the `grab` action with (right row) and without (left row) employing the action models and EBNN. In this initial stage of learning, when little data is yet available, the generalization bias from the pre-learned action models is apparent. Although none of the Q functions has yet converged, the Q functions learned using EBNN have a shape that is more correct, and which is unlikely to be guessed based solely on the few observed training points with no initial knowledge. For example, even after presenting six episodes the plain learning procedure predicts positive reward solely based upon the angle of the cup, whereas the combined EBNN method has already learned that grasping will fail if the cup is too far away. This information, however, is not represented in the training episodes (Figure 7), since there is no single example of an attempt to grasp a cup far away. It rather seems that the slopes of the model were "copied" into the target evaluation function. This illustrates the effect of the slopes in EBNN: The evaluation functions learned with EBNN discovered the correlation of the distance of the cup and the success of the `grab` action from the neural network action model. If these action models were learned in an earlier control learning tasks, there would be a significant synergy effect between them.

The EBNN results in this section are initial. They are presented because they indicate that task-independent knowledge, once learned, can be successfully transferred by EBNN when learning the grasping task. They are also presented since they evaluated EBNN in a real robot domain, unlike the results presented in [Mitchell and Thrun, 1993b], [Thrun and Mitchell, 1993]. However, thus far we did not collect enough

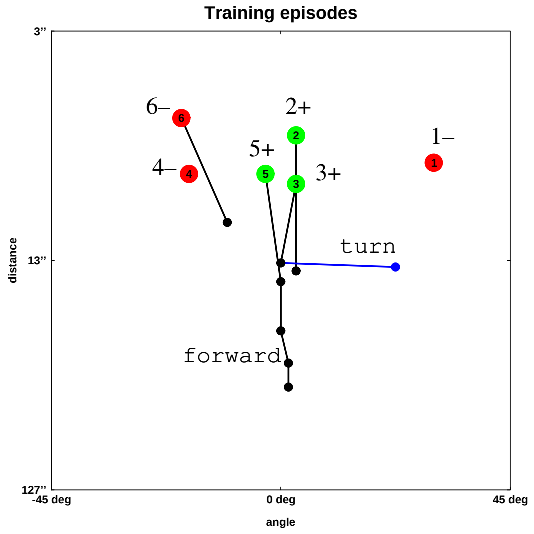

Figure 7 Six training episodes for learning control, labeled by 1 to 6. The horizontal axis measures the angle of the cup, relative to the robot's body, and the vertical axis measures the distance to the cup in a logarithmic scale. Successful grasps are labeled with "+", unsuccessful with "–". Notice that some of the episodes included forwarding and turning.

---

training data to learn a complete policy for approaching and grasping the cup with either method. Future research will characterize the synergy during the full course of learning until convergence. It will also experiment with families of related tasks, such as approaching and grasping different objects, including cups that lie on the side.

3.5 EBNN and Lifelong Robot Learning

What lesson does EBNN teach us in the context of lifelong agent learning? In this section we made the restricting assumption that all control learning problems of the robot agent play in the very same environment. If this is the case, any type of models of the robot and its environment are promising candidates for transferring task-invariant knowledge between the individual control learning problems. In EBNN, task-independent information is represented by neural network action models. These models bias generalization when learning control via the process of explaining and analyzing observed episodes. EBNN is a method for lifelong agent learning, since it learns and re-uses learned knowledge that is independent of the particular control learning problem at hand. Although the initial experiments described in this paper do not fully demonstrate this point, EBNN is able to efficiently replace real-world

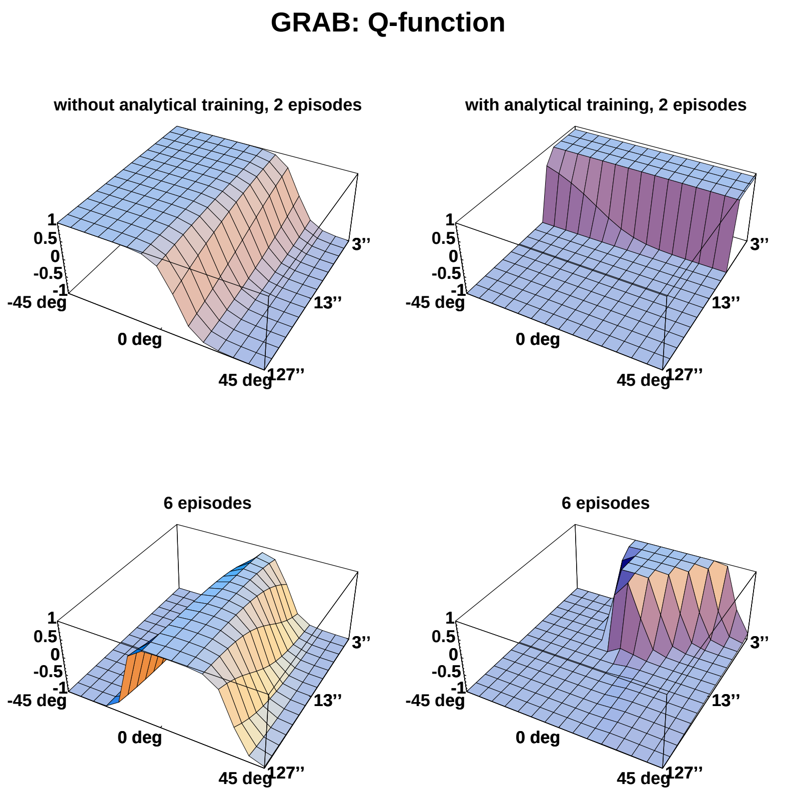

Figure 8 Evaluation functions for the action `grab` after presenting 2 (upper row) and 6 (lower row) training episodes, and without (left column) and with (right column) using the action model for biasing the generalization.

---

experimentation by previously learned bias. In related experiments with a simulated robot presented elsewhere [Mitchell and Thrun, 1993b], we observed a significant speed-up by a factor of 3 to 4 when pre-learned action models were employed.

It is important to mention that we expect EBNN to be even more efficient if the dimension of the state space is higher. This is because for each single observation EBNN extracts a d-dimensional slope vector from the environment, if d if the dimension of the input space. Our conjecture is that with accurate domain theories the generalization improvement scales linearly with the number of instance features (e.g., we would expect a three order of magnitude improvement for a network with 1000 input features), since each derivative extracted by EBNN can be viewed roughly as summarizing information equivalent to one new training example for each of the d input dimensions. This conjecture is roughly consistent with our experiments in the simulated robot domain. The relative performance improvement might increase even more if higher-order derivatives are extracted, which is not yet the case in EBNN. It is important to notice, however, that the quality of the action models is crucial for the performance of EBNN. With inaccurate action models, EBNN will not perform better than a purely inductive learning procedure. This does not surprise. Approaches to the lifelong learning problem will always be at most as efficient as non-transferring approaches the robot solves its first control learning problem in its life. The desired synergy effect occurs later, when the agent has learned an appropriate domain-specific bias, when it is more *"mature."*

4 Lifelong Learning in Multiple Environments

In the previous section we focussed on a certain type of lifelong robot learning problems, namely problems which deal with a single environment. We will now focus on a more general type of lifelong-learning scenarios, in which the environments differ for the individual control learning tasks. As pointed out in Section 2 , there are quite a few robot learning scenarios where a robot has to learn control in whole families of environments. For example, a vacuum cleaning robot might have to learn to clean different buildings. Alternatively, an autonomous vehicle might have to learn navigation in several types of terrain.

Multi-environment scenarios provide less possibilities for transferring knowledge. At a first glance, transferring knowledge seems hard, if the environment is not the same for all control learning tasks. But even in this type problems there are invariants that may be learned and used as a bias. The key observation is that all lifelong learning scenarios involve the same robot, the same effectors, the same sensors, although they might have to deal with a variety of environments and control learning tasks therein. Approaches to this type of lifelong learning problems thus aim at learning the characteristics of the sensors and effectors of the robot, as well as invariants in the environments, if there are any. The principle of learning and transferring task-independent knowledge is the same as in the previous section–just the type of knowledge that is transferred differs.

So far, there has been little systematic research on this general type of lifelong robot learning scenarios. In the remainder of this section we will not describe a general learning mechanism, but a particular approach to learning the characteristics of the robot's sensors and the environments. Using the HERO-2000 robot as an example, we will demonstrate how inverse sensor models can represent a knowledgeable bias which is independent of the particular environment at hand.

Figure 9 The robot explores an unknown environment. Note the obstacle in the middle of the laboratory. Our lab causes many malicious sonar values, and is a hard testbed for sonar-based navigation. For example, some of the chairs absorb sound almost completely, and are thus hard to detect by sonar.

4.1 Learning to Interpret Sensations

What kind of invariants can be learned across multiple environments? Two observations are crucial for the approach described here. First, the robot and its sensors are the same for each environment. Second, there might be regularities in the environments that can be learned as well. In this section we will describe a neural network approach to learning the characteristics of the robot's sensors, as well as those of typical indoor environments.

The task of the mobile HERO-2000 robot is to explore unknown buildings [Thrun, 1993]. Facing a new indoor environment such as the laboratory environment depicted in Figure 9, the robot has to wander around and to use its sensors to avoid collisions. In the exploration task the robot uses two of its sensors: A rotating sonar sensor is mounted on the head of the robot, as shown in Figure 2. The robot also monitors its wheels encoders to detect stuck or slipping wheels. Negative reward is received for collision which can be detected using the wheel encoders. Positive reward is received for entering regions where the robot has not been before. Initially, the robot does not possess any knowledge about its sensors and the environments it will face throughout its life. Sensations are uninterpreted 24-dimensional vectors of floats (sonar scans), along with a single bit that encodes the state of the wheels. In order to simplify learning, we assume that the robot has access to its x-y-θ coordinates

(b)

Figure 10 Task-independent knowledge: (a) sensor interpretation network $\mathcal{R}$ and (b) confidence network $\mathcal{C}$.

in a global reference frame (θ measures the orientation of the robot).[9] Navigation in unknown environments is clearly a lifelong robot learning problem. Initially, the robot has to experience collisions, since the initial knowledge does not suffice to prevent from them. Collisions will be penalized by negative reward. In order to transfer knowledge, the robot then has to learn how to interpret its sensors in order to prevent collisions. This knowledge is re-used for each environment the robot will face over its lifetime. After some initial experimentation, the robot should be able to maneuver in new, unknown world while successfully avoiding collisions with obstacles.

We will now describe a pair of networks which learn sensor-specific knowledge that can be re-used across multiple environment. The *sensor interpretation function*, denoted by $\mathcal{R}$, maps sensor information–in our case a sonar scan–to reward information. More specifically, $\mathcal{R}$ evaluates for arbitrary locations close to the robot the probability for a collision, based on a single sonar scan. Figure 10a shows $\mathcal{R}$. Input to the network is a vector of sonar values, together with the coordinates of the query point $(\Delta x, \Delta y)$ relative to the robot's local coordinate frame. The output of the network is 1, if the interpretation predicts a collision for this point, and 0, if the network predicts free-space. This function can be learned by standard supervised learning algorithms, if the robot keeps track of all sensor readings and of all locations where it collided (and where it did not collide). As the robot operates, it constructs maps that label occupied regions and free-space, which is used to form training examples for the sensor interpretation network $\mathcal{R}$. As usual, the robot uses Back-propagation to learn $\mathcal{R}$.

In order to prevent the precious robot hardware from real-world collisions, we

[9] If the robot wheels are perfect, this x-y-θ position can be calculated internally by dead reckoning. The robot at hand, however, is not precise enough, and after 10 to 20 minutes of operation the real coordinated usually deviate significantly from the internal estimates. An approach to compensating such control errors is briefly described in [Thrun, 1993].

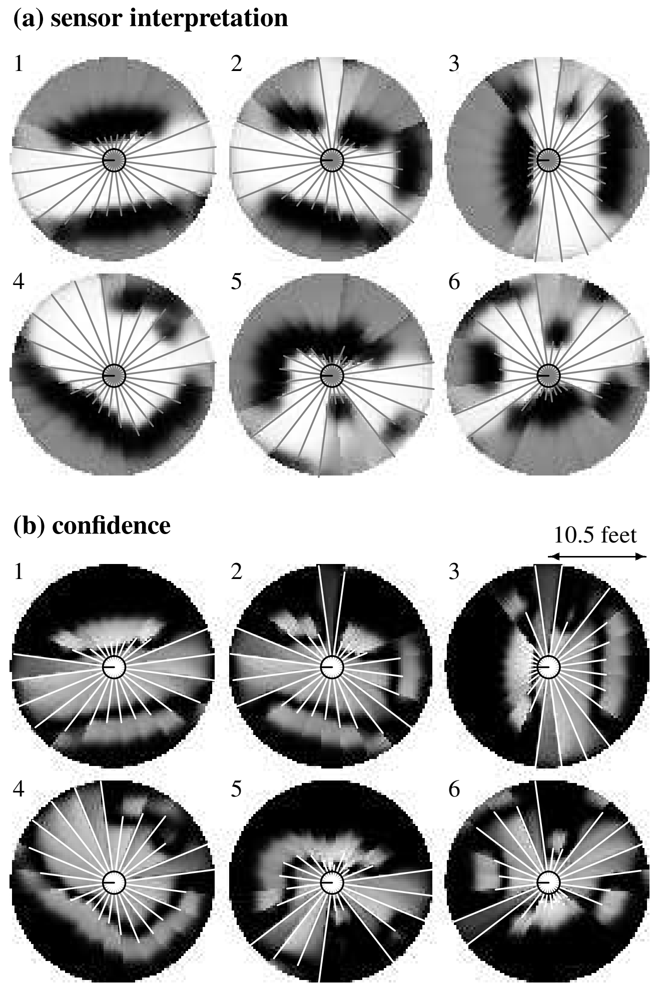

Figure 11 Capturing domain knowledge in the networks $\mathcal{R}$ and $\mathcal{C}$. (a) Sensor interpretations and (b) confidence values are shown for the following examples: 1. hallway, 2. hallway with open door. 3. hallway with human walking by, 4. corner of a room, 5. corner with obstacle, 6. several obstacles. Lines indicate sonar measurements (distances), and the region darkness represents in (a) the expected collision reward for surrounding areas (dark values indicate negative reward), and in (b) the confidence level (dark values indicate low confidence).

---

designed a robot simulator and used it for generating the training patterns for the interpretation network. In the simulator the whole robot environment is known, and training examples that map sensations to occupancy information can be extracted easily. In a few minutes the simulated robot explored its simulated world, collecting

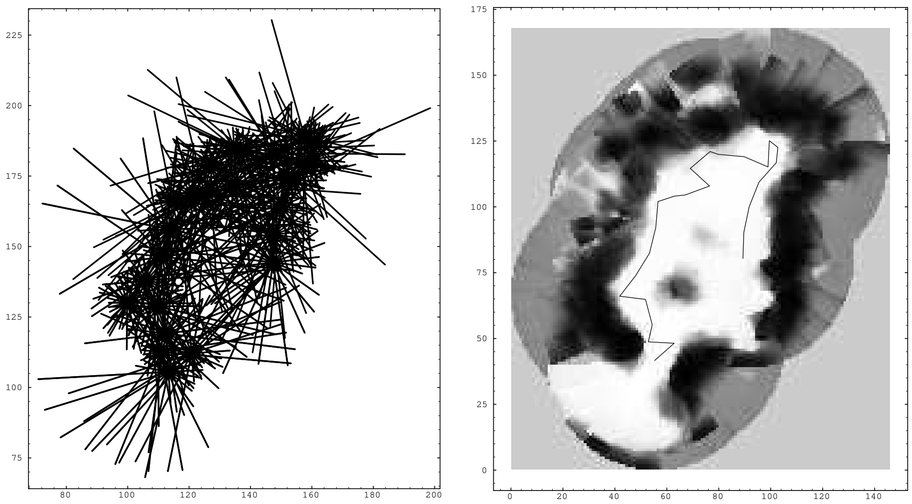

Figure 12 Map building: (a) Raw, noisy sensor input on an exploration path with 27 measurements. (b) Resulting model, corresponding to the lab shown in Figure 1b (lab doorway is toward bottom left). The path of the robot is also plotted (from right to left), demonstrating the exploration of the initially unknown lab. In the next steps, the robot will pass the door of the lab and explore the hallway.

---

a total of 8 192 training examples. Six examples for sensor interpretation using $\mathcal{R}$ are shown in Figure 11a. The circle in the center represents the robot, and the lines orthogonal to the robot represent distances sensed by the sonars. The probability of negative reward, as predicted by the $\mathcal{R}$, is displayed in the circular regions around the robot: The darker the region, the higher the probability of collision. As can be seen from this figure, the network $\mathcal{R}$ has successfully learned how to interpret sonar signals. If sonar values are small (meaning that the sonar signal bounced back early), the network predicts an obstacle nearby. Likewise, large value readings are interpreted as free-space. The network has also learned invariants in the training environments. For example, "typical" sizes of walls are known. In Figure 11a-1, for example, the robot predicts a long obstacle of a certain width, but behind this obstacle it predicts a considerably low probability for collision. This prediction is surprising, given that sonar sensors cannot "see through obstacles." In the training environments, however, regions behind walls happened to be free fairly often, which explains the X-ray predictions by the network. This provides clear evidence that the sensor interpretation network does not only represent knowledge about the robot's sensors, but also knowledge about certain invariants of the environments at hand.

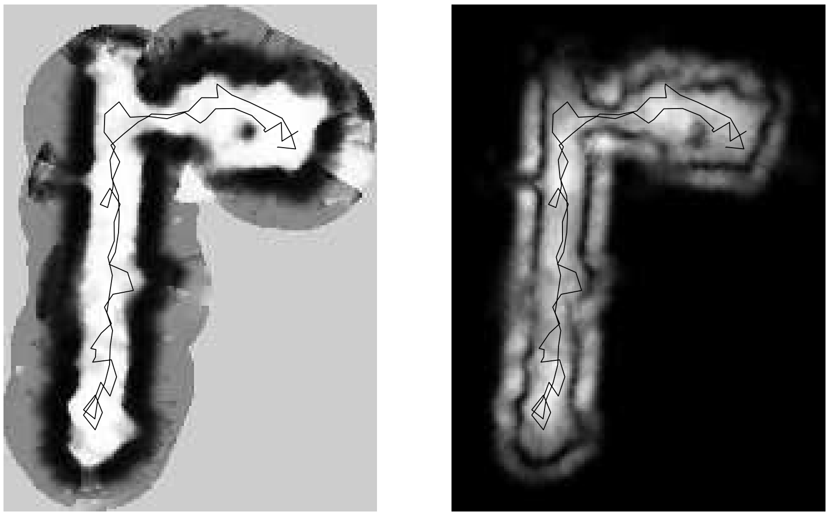

Figure 13 Lab and floor: (a) model, and (b) confidence map.

4.2 Building Maps

We will now motivate the need for a second network, the so-called *confidence network* $\mathcal{C}$, which is related to $\mathcal{R}$. As can bee seen from Figure 11a, a single sonar scan can be used to build a small local map around the robot. If the robot moves around and takes several readings, the resulting interpretations can be combined to form a *map*[10] of the world. However, there will be points in the world where the interpretations from multiple sensations will disagree. There are two main reasons for conflicting interpretations: First, sonar sensors, like any sensor, are noisy devices. They often fail to detect certain objects such as smooth surfaces or objects with absorbing surfaces like some of our office chairs. Second, sonars are blind for areas behind obstacles. Hence interpretations for regions behind an obstacle will be inaccurate–they truly reflect the average probability of occupancy (the *prior*). This observation makes it necessary to design a mechanism that resolves conflicts between different predictions.

We will approach this problem by explicitly modeling the reliability of the interpretations, and using the reliability as weighting factor when combining multiple interpretations. More specifically, consider a testing phase for the interpretation network $\mathcal{R}$. For some of the testing examples, $\mathcal{R}$ will manage to predict the target value closely. For others, however, there will be a significant residual error for the

[10] See [Moravec, 1988] and [Elfes, 1987] for related approaches to map building and robot navigation.

Figure 14 Map compiled during the first AAAI autonomous robot competition in San Jose, July 1992. The map reflects the knowledge of the robot after Stage 3 of the competition. In this and the previous stage, the robot had to find and approach ten visually marked poles, while avoiding collision with obstacles. Lines mark the boundary of the competition area. The map is somehow inaccurate since it was built on two separate days, and the locations of the robot and the obstacles were not quite identical. We compensated for such inaccuracies by decaying the confidence over time.

---

reasons given above. This error is used to train a second network $\mathcal{C}$, which is called *confidence network*. The input to $\mathcal{C}$ is the same as the input to $\mathcal{R}$. The target output is the (normalized) prediction error of $\mathcal{R}$. After training, $\mathcal{C}$ thus predicts the expected deviation of the sensor interpretations, denoted by $\mathcal{C}(s, \Delta x, \Delta y)$. The confidence into these interpretation $\mathcal{R}(s, \Delta x, \Delta y)$ is thus given by $-\ln \mathcal{C}(s, \Delta x, \Delta y)$. When multiple sensor interpretations are combined, the individual interpretation values are averaged, weighted by their confidence value. Figure 11b shows confidence values for the interpretations showed in Figure 11a. Here dark regions correspond to low confidence. Likewise, light regions indicate high confidence. As can be seen from these examples, the confidence in regions behind obstacles is generally low. Low confidence is also predicted for boundary regions between free-space and obstacles, This does not surprise, since sonar values detect objects in a 15° cone. They do not tell *where* in the cone an object was found. Consequently, the fine-structure of objects is hard to predict. Similar to the sensor interpretation network, the confidence network represents knowledge about the sensors and the environments of the robot that can be transferred across environments. It is important to notice that both networks, $\mathcal{R}$ and $\mathcal{C}$, represent learned knowledge that is independent of the particular environment at hand. These networks act as a bias in the modeling process, when the robot constructs an internal model of a new environment.

The Figures 12 to 14 show some example maps that were obtained for different environments. We used the networks $\mathcal{R}$ and $\mathcal{C}$ to find models of our lab (9). A simple non-adaptive algorithm was used for exploration that basically planned minimal-cost plans to the closest unexplored region. Models of the lab and the hallway are shown in Figures 12 and 13. Although both networks were trained in simulation, they successfully prevented the robot from colliding with obstacles. The same networks $\mathcal{R}$ and $\mathcal{C}$ were used on a autonomous robot competition, that was held during the AAAI conference in July 1992 in San Jose. Here the task and environment differed from the environments the robot had seen previously: The environment was a large arena filled with paper boxes, and the task was to find and to navigate to 10 visually marked poles. The robot had no information about the location of the obstacles and the poles. It had to explore and model the environment by itself. The final implementation (we named the robot "Odysseus") was far more complex than what is described here. Odysseus' navigation was map-based, and the adaptive map building procedure was a component of Odysseus' control. The robot employed the same networks that were generated by the simulator and tested in our experiments in the lab. After a total of 40 minutes operation during two separate stages of the competition the robot produced the map shown in Figure 14. The networks produced maps that were accurate enough to protect the robot from any collision with an obstacle. They also allowed the robot to find close-to-optimal paths to the goal locations.

4.3 Sensor Interpretation and Lifelong Robot Learning

What is the contribution of this approach to the problem of lifelong agent learning? Of course, learning sensor interpretation does not necessarily have to be viewed as a method for learning bias, and the presented method is by no means a general learning scheme for lifelong learning problems. However, for the purpose of this paper we will characterize map building in terms of lifelong robot learning. Obviously in each new environment the robot clearly has to learn control. This is because initially, when the robot faces a new, unknown environment, its knowledge does clearly not suffice to generate actions that maximize reward. The robot then gradually learns a policy for action generation step-by-step, and the internal maps provide the freedom for learning, the knowledge "gaps" filled during the course of interaction with the world. Building internal two-dimensional occupancy maps, together with the static planning routine that generates action using these maps, is learning control. Thus, learning neural network sensor interpretations offers a promising perspective for research in lifelong robot learning in multiple environments. Although both the environments and the goals differed for the individual robot control tasks in our experiments, we have demonstrated that knowledge transfer could drastically reduce real-world experimentation. The robot did know about collisions in both real-world environments without ever having experienced one, solely based on previously learned knowledge. We believe that methods which acquire and utilize models of the sensors, effectors (not demonstrated here), and invariants in the environments are promising candidates for efficient robot learning in more complex lifelong learning scenarios.

5 Related Work

Approaches to knowledge transfer in the Lifelong Learning Problems can be viewed as techniques for acquiring function approximation bias. Various researchers have noted the importance of learning bias and transferring knowledge across multiple robot control learning tasks. They can roughly be grouped into the following categories:

1. **Learning models.** Action models are perhaps the most straightforward way to learn and transfer task-independent knowledge, if all individual control learning tasks deal with a single environment. Approaches that utilize action models differ in the type of action models they employ, and the way the action models are used to bias learning control. Sutton [Sutton, 1990] presents a system that learns action models, like EBNN. He uses these models for synthesizing hypothetical experiences that refine the control policy. In his experiments he found a tremendous speedup in learning control when using these action models. EBNN differs from this approach in that it uses its action models for explaining real-world observations, and in that provides a mechanism to recover from errors in the action models. Lin [Lin, 1992a] describes a mechanism where past experience is memorized and repeatedly replayed when learning control. The collection of past experience forms a non-generalizing action model. Lin also reports significant speedups when using his replay mechanism. As mentioned above, experience replay was also used for neural network training in EBNN. Thus far, there has been little research on learning models that can act as a bias in lifelong learning problems with multiple robot environments.
2. **Learning behaviors and abstractions.** A second way of learning and transferring knowledge across tasks are behaviors. Behaviors are controllers (policies) with low complexity–Often the term behavior refers to reactive controllers. Reinforcement learning, for example, can be viewed as a technique for learning behaviors. Behaviors form abstract action spaces, since the basic actions of the robot might be replaced by the action of invoking a behavior. Thus, with appropriate behaviors abstract action spaces can be formed, and hierarchies of actions can be identified. Learning behaviors accelerates learning control by restricting the search space of all possible policies, mainly for two reasons. First, the number of behaviors is often smaller than the number of actions. Second, behaviors

typically are selected for longer periods of time. The latter argument is usually more important and provides a stronger bias for learning control.
Singh [Singh, 1992b], [Singh, 1992a] reports a technique to learn complex tasks by first learning controllers for simple tasks based on reinforcement learning. These controllers represent reactive behaviors. The high-level policy is learned in the abstract action space formed by the low level behaviors. In order to represent even the high-level controller as a purely reactive function, Singh makes several restricting assumptions on the type of tasks and sensor information. In his doctoral thesis, Lin [Lin, 1992b] describes a related scheme for learning behaviors, action hierarchies and abstraction. He assumes that a human instructor teaches a robot a set of elemental behaviors which suffice for all tasks which the robot will face over its lifetime. Unlike Singh, his approach does not guarantee that optimal controller can be learned in the limit. Recently, Dayan and Hinton [Dayan and Hinton, 1993] proposed a system that uses a pre-given hierarchical decomposition to learn control on different levels of abstraction. Each level of abstraction differs in the grain-size of the sensory information, resulting in differently specialized controllers. Since their system learns reactive controllers on each level using reinforcement learning algorithms, it is unclear for what type of problems this procedure will learn successfully, since essential information may be missed when providing incomplete sensor information to purely reactive controllers.

3. **Learning inductive function approximation bias.** Another, more straightforward approach to learning and transferring knowledge is to learn the inductive bias of the function approximators used for learning control directly. Atkeson [Atkeson, 1991] presents a scheme for learning distance measures for instance-based, local approximation schemes. In his algorithm, scaling factors are learned that allows to weight different input features differently. Sutton [Sutton, 1992] reports a family of learning schemes that allow to learn inductive bias similar to Kalman filters [Kalman, 1960]. Although he did not describe his methods in the context of learning control, his research has been motivated by transferring knowledge across multiple control learning tasks.
4. **Learning representations.** Representations, together with inductive bias, determine the way a function approximator generalizes from examples. Many researchers have focussed on learning appropriate representations in order to learn bias. For example, Pratt [Pratt, 1993] describes several approaches that allow to re-use learned representations in hidden units of neural networks. Although she could empirically demonstrate that this transfer could significantly reduce the number of training epochs required for the convergence of the Back-propagation algorithm, she only found occasional improvements in the generalization. A similar technique is reported by Sharkey and Sharkey [Sharkey and Sharkey, 1992]. Some researchers have studied knowledge transfer if several tasks are learned simultaneously. For example, Suddarth and Kergosien [Suddarth and Kergosien, 1990] demonstrated that multiple learning tasks of certain types can successfully guide and improve generalization. In his approach, he gives *hints* to neural

networks in form of additional output units that learn a closely related task. These hints constrain the internal representation developed by the network. In a more general way, Caruana [Caruana, 1993] recently proposed to learn whole collections of tasks in parallel, using a shared internal representation. He conjectures that multi-task learning will make neural network learning algorithms scale to more complex learning tasks.

Both approaches to the lifelong learning problem described in this paper fall into the first category. In EBNN, the robot learns action models which bias generalization in the evaluation functions. Sensor interpretation functions are inverse models of the sensors and the environments of the robot. In the robot exploration tasks, the robot thus learns models of itself and typical aspects of its environments, which bias the construction of the individual maps. As pointed out earlier, action models are well-suited if the environment is the same for all learning tasks, whereas sensor models are appropriate for lifelong agent learning in multiple environments.

6 Conclusion

In this paper we have presented a lifelong learning perspective for autonomous robots. We propose not to study robot learning problems in isolation, but in the context of the multitude of learning problems that a robot will face over its lifetime. Lifelong robot learning opens the opportunity for transfer of learned knowledge. This knowledge may be used as a bias when learning control. Although control learning methods that allow the transfer of knowledge are more complex as most algorithms that solve isolated control learning problems, robot learning itself becomes easier. Robots that memorize and transfer knowledge rely less on real-world experimentation and thus learn faster. This is because previously learned knowledge may act as a knowledgeable bias that may partially replace the pure syntactic bias of inductive learning algorithms.

We have demonstrated with two concrete approaches the potential synergy effect of knowledge transfer. These approaches addressed two main types of lifelong agent learning scenarios, namely those that are defined in a single environment, and those that are not. We strongly believe that knowledge transfer is essential for scaling robot learning algorithms to more realistic and complex domains. Exploiting previously learned knowledge simplifies learning control. These results support our fundamental claim that learning becomes easier, if it is embedded into a lifelong learning context.

Acknowledgment

We thank the CMU Robot Learning Group and the Odysseus team at CMU for invaluable discussion that contributed to this research. We also thank Ryusuke Masuoka

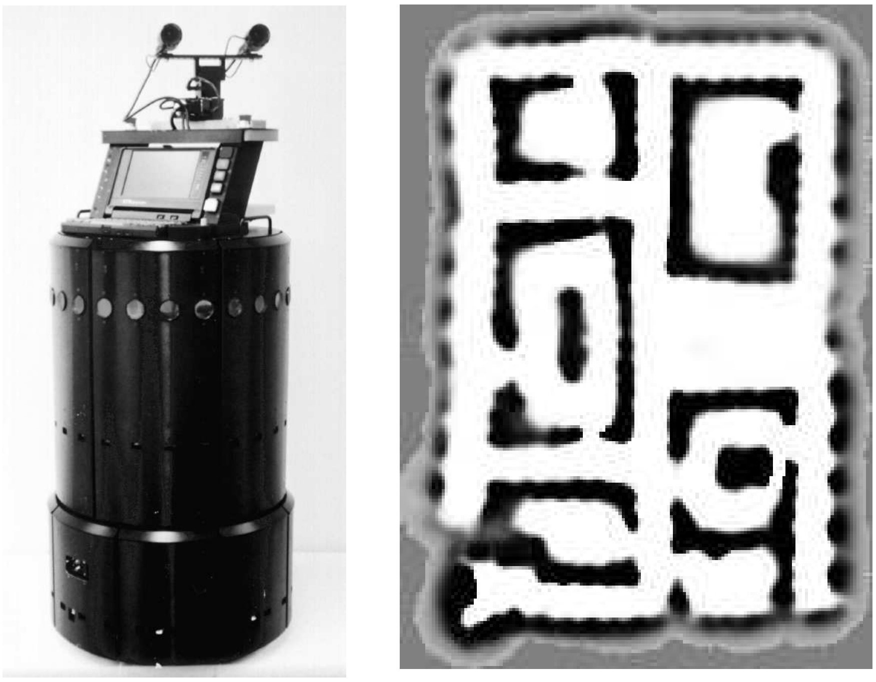

Figure 15 (a) The University of Bonn robot "Rhino" (manufactured by Real World Interface, Inc.), (b) Map (approximately 20 $\times$ 30 meters) constructed by Rhino at the AAAI-94 robot competition, using the technique described in this paper.

for his invaluable help in refining EBNN.

This research was sponsored in part by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597 and by a grant from Siemens Corporation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government or Siemens Corp.

Addendum

Since this paper was submitted, EBNN was successfully applied to a variety of real-world learning tasks. In [Mitchell and Thrun, 1995, Thrun, 1994, Thrun, 1995a], result of applying EBNN to mobile robot navigation using the CMU Xavier robot are reported. EBNN has also been applied to robot perception [Mitchell *et al.*, 1994], [O'Sullivan *et al.*, to appear], object recognition [Thrun and Mitchell, 1994] and the game of chess [Thrun, 1995b]. In [Thrun and Mitchell, 1994], a definition of the lifelong learning problem in the context of supervised learning can be found.

The approach to interpreting sonar sensors for building occupancy maps reported in Sect. 4 has been, with slight modifications, successfully employed in the University of Bonn's entry "Rhino" at the 1994 AAAI mobile robot competition [Buhmann *et al.*, to appear]. Currently, maps are routinely built for large indoor areas.

References

[Atkeson, 1991] Christopher A. Atkeson. Using locally weighted regression for robot learning. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 958–962, Sacramento, CA, April 1991.

[Bachrach and Mozer, 1991] Jonathan R. Bachrach and Michael C. Mozer. Connectionist modeling and control of finite state systems given partial state information. 1991.

[Barto *et al.*, 1989] Andrew G. Barto, Richard S. Sutton, and Chris J. C. H. Watkins. Learning and sequential decision making. Technical Report COINS 89-95, Department of Computer Science, University of Massachusetts, MA, September 1989.

[Barto *et al.*, 1990] Andrew G. Barto, Richard S. Sutton, and Chris J. C. H. Watkins. Learning and sequential decision making. In M. Gabriel and J.W. Moore, editors, *Learning and Computational Neuroscience*, pages 539–602, Cambridge, Massachusetts, 1990. MIT Press.

[Barto *et al.*, 1991] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Real-time learning and control using asynchronous dynamic programming. Technical Report COINS 91-57, Department of Computer Science, University of Massachusetts, MA, August 1991.

[Bellman, 1957] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[Brooks, 1989] Rodney A. Brooks. A robot that walks; emergent behaviors from a carefully evolved network. *Neural Computation*, 1(2):253, 1989.

[Buhmann *et al.*, to appear] Joachim Buhmann, Wolfram Burgard, Armin B. Cremers, Dieter Fox, Thomas Hofmann, Frank Schneider, Jiannis Strikos, and Sebastian Thrun. The mobile robot Rhino. *AI Magazine*, 16(1), to appear.

[Bylander, 1991] Tom Bylander. Complexity results for planning. In *Proceedings of IJCAI-91*, pages 274–279, Darling Habour, Sydney, Australia, 1991. IJCAI, Inc.

[Canny, 1987] John Canny. *The Complexity of Robot Motion Planning*. The MIT Press, Cambridge, MA, 1987.

[Caruana, 1993] Richard Caruana. Multitask learning: A knowledge-based of source of inductive bias. In Paul E. Utgoff, editor, *Proceedings of the Tenth*

International Conference on Machine Learning, pages 41–48, San Mateo, CA, 1993. Morgan Kaufmann.

[Chrisman, 1992] Lonnie Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinction approach. In *Proceedings of 1992 AAAI Conference*, Menlo Park, CA, July 1992. AAAI Press / The MIT Press.

[Dayan and Hinton, 1993] Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 5*, San Mateo, CA, 1993. Morgan Kaufmann.

[DeJong and Mooney, 1986] Gerald DeJong and Raymond Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176, 1986.

[Elfes, 1987] Alberto Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, June 1987.

[Jordan, 1989] Michael I. Jordan. Generic constraints on underspecified target trajectories. In *Proceedings of the First International Joint Conference on Neural Networks, Washington, DC*, San Diego, 1989. IEEE TAB Neural Network Committee.

[Kalman, 1960] R.E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering*, 82:35–45, 1960.

[Kuipers and Byun, 1990] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. Technical report, Department of Computer Science, University of Texas at Austin, Austin, Texas 78712, January 1990.

[Lin and Mitchell, 1992] Long-Ji Lin and Tom M. Mitchell. Memory approaches to reinforcement learning in non-markovian domains. Technical Report CMU-CS-92-138, Carnegie Mellon University, Pittsburgh, PA, 1992.

[Lin, 1992a] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8, 1992.

[Lin, 1992b] Long-Ji Lin. *Self-supervised Learning by Reinforcement and Artificial Neural Networks*. PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 1992.

[Maes, 1991] Pattie Maes, editor. *Designing Autonomous Agents*. The MIT Press (and Elsevier), Cambridge, MA, 1991.

[Mahadevan and Connell, 1991] Sridhar Mahadevan and Jonathan Connell. Scaling reinforcement learning to robotics by exploiting the subsumption architecture. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 328–332, 1991.

[Mel, 1989] Bartlett W. Mel. Murphy: A neurally-inspired connectionist approach to learning and performance in vision-based robot motion planning. Technical Report CCSR-89-17A, Center for Complex Systems Research Beckman Institute, University of Illinois, 1989.

[Mitchell and Thrun, 1993a] Tom M. Mitchell and Sebastian Thrun. Explanation based learning: A comparison of symbolic and neural network approaches. In

Paul E. Utgoff, editor, *Proceedings of the Tenth International Conference on Machine Learning*, pages 197–204, San Mateo, CA, 1993. Morgan Kaufmann.

[Mitchell and Thrun, 1993b] Tom M. Mitchell and Sebastian Thrun. Explanation-based neural network learning for robot control. In S. J. Hanson, J. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 287–294, San Mateo, CA, 1993. Morgan Kaufmann.

[Mitchell and Thrun, 1995] Tom M. Mitchell and Sebastian Thrun. Learning analytically and inductively. In Steier and Mitchell, editors, *Mind Matters: A Tribute to Allen Newell*. Lawrence Erlbaum Associates Publisher, 1995.

[Mitchell *et al.*, 1986] Tom M. Mitchell, Rich Keller, and Smadar Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[Mitchell *et al.*, 1994] Tom M. Mitchell, Joseph O'Sullivan, and Sebastian Thrun. Explanation-based learning for mobile robot perception. In *Workshop on Robot Learning, Eleventh Conference on Machine Learning*, 1994.

[Moore, 1990] Andrew W. Moore. *Efficient Memory-based Learning for Robot Control*. PhD thesis, Trinity Hall, University of Cambridge, England, 1990.

[Moravec, 1988] Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

[Mozer and Bachrach, 1989] Michael C. Mozer and Jonathan R. Bachrach. Discovering the structure of a reactive environment by exploration. Technical Report CU-CS-451-89, Dept. of Computer Science, University of Colorado, Boulder, November 1989.

[Munro, 1987] Paul Munro. A dual backpropagation scheme for scalar-reward learning. In *Ninth Annual Conference of the Cognitive Science Society*, pages 165–176, Hillsdale, NJ, 1987. Cognitive Science Society, Lawrence Erlbaum.

[O'Sullivan *et al.*, to appear] Joseph O'Sullivan, Tom M. Mitchell, and Sebastian Thrun. Explanation-based neural network learning from mobile robot perception. In Katsushi Ikeuchi and Manuela Veloso, editors, *Symbolic Visual Learning*. Oxford University Press, to appear.

[Pomerleau, 1989] Dean A. Pomerleau. ALVINN: an autonomous land vehicle in a neural network. Technical Report CMU-CS-89-107, Computer Science Dept. Carnegie Mellon University, Pittsburgh PA, 1989.

[Pratt, 1993] Lori Y. Pratt. Discriminability-based transfer between neural networks. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 5*, San Mateo, CA, 1993. Morgan Kaufmann.

[Rivest and Schapire, 1987] Ronald L. Rivest and Robert E. Schapire. Diversity-based inference of finite automata. In *Proceedings of Foundations of Computer Science*, 1987.

[Rumelhart *et al.*, 1986] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing. Vol. I + II*. MIT Press, 1986.

[Samuel, 1959] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal on research and development*, 3:210–229, 1959.

[Schwartz *et al.*, 1987] Jacob T. Schwartz, Micha Scharir, and John Hopcroft. *Planning, Geometry and Complexity of Robot Motion*. Ablex Publishing Corporation, Norwood, NJ, 1987.

[Sharkey and Sharkey, 1992] Noel E. Sharkey and Amanda J.C. Sharkey. Adaptive generalization and the transfer of knowledge. In *Proceedings of the Second Irish Neural Networks Conference*, Belfast, 1992.

[Simard *et al.*, 1992] Patrice Simard, Bernard Victorri, Yann LeCun, and John Denker. Tangent prop – a formalism for specifying selected invariances in an adaptive network. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 895–903, San Mateo, CA, 1992. Morgan Kaufmann.

[Singh, 1992a] Satinder P. Singh. The efficient learning of multiple task sequences. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 251–258, San Mateo, CA, 1992. Morgan Kaufmann.

[Singh, 1992b] Satinder P. Singh. Transfer of learning by composing solutions for elemental sequential tasks. *Machine Learning*, 8, 1992.

[Suddarth and Kergosien, 1990] Steven C. Suddarth and Y. L. Kergosien. Rule-injection hints as a means of improving network performance and learning time. In *Proceedings of the EURASIP Workshop on Neural Networks*, Sesimbra, Portugal, Feb 1990. EURASIP.

[Sutton, 1984] Richard S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, Department of Computer and Information Science, University of Massachusetts, 1984.

[Sutton, 1988] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3, 1988.

[Sutton, 1990] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning, June 1990*, pages 216–224, San Mateo, CA, 1990. Morgan Kaufmann.

[Sutton, 1992] Richard S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proceeding of Tenth National Conference on Artificial Intelligence AAAI-92*, pages 171–176, Menlo Park, CA, July 1992. AAAI, AAAI Press/The MIT Press.

[Tan, 1991] Ming Tan. Learning a cost-sensitive internal representation for reinforcement learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 358–362, 1991.

[Thrun and Mitchell, 1993] Sebastian Thrun and Tom M. Mitchell. Integrating inductive neural network learning and explanation-based learning. In *Proceedings of IJCAI-93*, Chamberry, France, July 1993. IJCAI, Inc.

[Thrun and Mitchell, 1994] Sebastian Thrun and Tom M. Mitchell. Learning one more thing. Technical Report CMU-CS-94-184, Carnegie Mellon University, Pittsburgh, PA 15213, September 1994.

[Thrun, 1992] Sebastian Thrun. The role of exploration in learning control. In David A. White and Donald A. Sofge, editors, *Handbook of intelligent control: neural, fuzzy and adaptive approaches*. Van Nostrand Reinhold, Florence, Kentucky 41022, 1992.

[Thrun, 1993] Sebastian Thrun. Exploration and model building in mobile robot domains. In *Proceedings of the ICNN-93*, pages 175–180, San Francisco, CA, March 1993. IEEE Neural Network Council.

[Thrun, 1994] Sebastian Thrun. A lifelong learning perspective for mobile robot control. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, September 1994.

[Thrun, 1995a] Sebastian Thrun. An approach to learning mobile robot navigation. *Robotics and Autonomous Systems*, 1995. (in press).

[Thrun, 1995b] Sebastian Thrun. Learning to play the game of chess. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, San Mateo, CA, 1995. Morgan Kaufmann. (to appear).

[Watkins, 1989] Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, England, 1989.

[Whitehead and Ballard, 1991] Steven D. Whitehead and Dana H. Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7:45–83, 1991.