Context-Sensitive Planning for Autonomous Vehicles

by

David Vengerov

B.S., Mathematics (1997)

Massachusetts Institute of Technology

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1998

© 1998 by David Vengerov, all rights reserved

The author hereby grants to MIT permission to reproduce and to distribute publically paper and electronic copies of this thesis document in whole or in part.

Signature of Author _____

Department of Electrical Engineering and Computer Science, June 1998

Certified by _____

Robert C. Berwick, Thesis Supervisor, MIT AI Laboratory

Accepted by _____

Arthur C. Smith, Chair, Department Committee on Graduate Students

LIBRARIES

To Elena,

With whom I was reborn…

Context-Sensitive Planning for Autonomous Vehicles

by

David Vengerov

Submitted to the Department of Electrical Engineering and Computer Science on May 28, 1998 in Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical Engineering and Computer Science

Abstract

The need for intelligent autonomous agents is growing in all areas of science and industry. This thesis addresses the problem of planning by an agent deployed to complete a mission in a complex, uncertain, and nonstationary environment. Agents operating in such environments need the capability to plan and act at different levels of abstraction. The level of abstraction at which an agent's actions need to be implemented depends on the *context* present in the environment, which might be difficult to recognize due to noise and partial observability. A favorable context may allow for more goal-oriented behavior, while an unfavorable context may demand more tactical behavior. Such contexts change over time, and for this reason behavior and planning policies need to be continuously evaluated and updated to fit the current context. This thesis describes, implements, and tests an architecture for evaluating and updating policies in such environments. The backbone of the architecture is a reinforcement learning (RL) component, which controls interactions with the agent's environment. A neural network component performs context-sensitive policy evaluation based on targets provided by the RL component. Finally, fuzzy logic rules perform policy updating based on values provided by the neural network as well as on domain-specific mission constraints. The ideas developed in this thesis are tested on the example of a survey mission for autonomous helicopters.

Thesis Supervisor: Robert C. Berwick, MIT Artificial Intelligence Laboratory

Title: Professor of Computer Science

Acknowledgments

First of all, I would like to thank my thesis supervisor, Professor Robert C. Berwick, for his support and his faith in me finishing this thesis. Without him, this work would literally be impossible. I would also like to thank Michael J. Ricard from the Draper Laboratory for providing me with valuable technical information on autonomous vehicles. This research has been made possible by C. S. Draper Laboratory that funded my education while I was working on this thesis.

Then, I would like to thank my father, Alexander Vengerov, for constantly steering my on the right path, in this thesis and in life. I would like to give my sweetest thanks to my fiancée, Elena Konstantinova, without whom I wouldn't be inspired to get half of ideas in this thesis.

Finally, I would like to thank my friends and roommates, Karén Pivazyan and Timothy Chklovski, for their willingness to listen to my ideas and to put them back to Earth.

Table of Contents

Those who will not reason
Perish in the act:
Those who will not act
Perish for that reason.
– W. H. Auden (1966)

Chapter 1

Introduction

1.1 The Planning Problem for Autonomous Agents

An autonomous agent can be described as any versatile adaptive system that performs diverse behaviors in order to achieve multiple goals. For concreteness, this thesis will adopt the example of autonomous vehicles, analyzed, as they are desired to operate in challenging external environments. The planning task for such a vehicle is to choose some future course of action in order to successfully accomplish its mission in a constrained environment. This is the primary capability required of all autonomous agents, and its adequate solution can lead to numerous scientific and industrial advances.

The goal of "mission success" in a planning problem often depends on achieving several individual goals imposed on the mission from independent sources, at different levels of abstraction. Examples of such goals are: speeding up the mission to conserve fuel, avoiding oncoming obstacles, staying within certain bounds of a long-term course, taking pictures at appropriate resolution for recognizing search objects. However, these goals usually cannot be achieved at the level of satisfaction required by each source. The vehicle always has to trade off the level of goal satisfaction with respect to different sources. The above trade-offs can be reasonably evaluated only by considering extended periods of plan execution [Pell et. al., 1996a, 1996b].

There are several sources of constraints placed on the action plans of autonomous vehicles. Virtually all resources of a vehicle are limited and have to be allocated effectively for goal-achievement. Fuel and electrical energy most critically limit mission time span. The planner has to reason about resource usage in generating plans, but because of run-time uncertainty the resource constraints must be enforced as a part of plan execution. In addition to resource constraints, the environment can impose logical constraints on a vehicle's actions. For example, an autonomous helicopter might be constrained to fly lower than a certain altitude so that it wouldn't be detected by enemy radars located nearby. Finally, the vehicle itself has limited capabilities, which must also be accounted for by the planner.

1.2 Environment Characteristics

The above planning problem will be analyzed in this thesis for autonomous vehicles operating in complex, uncertain, and nonstationary environments. Explanation of these environment characteristics and their implications for operations of autonomous vehicles is discussed below.

Environment complexity implies a very large number of possible situations that can be encountered during mission execution. The situations may depend on ground terrain, weather conditions, type and density of obstacles, danger level due to hostile activities, etc. *Environment dynamics* implies changes in the above conditions as the vehicle is moving between the areas. This dynamics is usually stochastic and partially known. *Environment nonstationarity* further implies unpredictable drift in the above dynamics.

Besides the lack of knowledge about future situations in the environment, autonomous vehicles also have incomplete information about the current situation, which is commonly termed partial observability. A special case of partial observability is that of hidden contexts in the environment, which affect the outcomes of actions taken by the agent. These contexts are represented by all functions or dependencies that affect the outcomes of agent's actions. An example of

such a dependency is the probability distribution describing the chance of collision with unknown obstacles as a function of vehicle's speed. These dependencies can be hidden from the agent because of vehicle's sensor limitations or its lack of background information about the environment needed to estimate such dependencies. It is crucial for the agent to be able to recognize these contexts and act appropriately in each one. For example, rapid motion is very undesirable in the context of poor visibility and multiple obstacles. On the other hand, fast motion could just speed up the achievement of mission goals in the context of poor visibility but a clear operating space.

The problems of environment uncertainty and partial observability are more acute in bad weather conditions such as strong winds or bad visibility and radar penetration due to rain or dust in the air. Also, the dynamics of moving obstacles such as other ships or aircraft becomes more difficult to model. The data available to the vehicle might originate from its own sensors or from communication both with other vehicles and ground-based stations. These sources provide information of varying degrees of quality and reliability. Intelligent processing of data from several sources has the potential of conveying more relevant information than data from a single sensor. [Doyle, 1996].

The problems of environment uncertainty and partial observability are exacerbated by strict limitations on the number of sensors available on the vehicle. Addition of sensors implies added mass, power, cabling, and up front engineering time and effort. Each sensor must add a clear value to the mission to be justified for inclusion. Furthermore, sensors are typically no more reliable than the vehicle hardware, making it that much more difficult to deduce the true state of the vehicle [Pell et. al., 1996a, 1996b].

In the view of the above characteristics of real-world environments, the following issues need to be addressed by a planning architecture for autonomous vehicles:

- Continuous policy evaluation

- An integrated policy updating mechanism
- No knowledge of the future environment states
- Continuous state and action spaces
- Noisy, nonlinear, nonstationary, and unknown state transition function
- Noisy policy feedback from environment
- Evolving goal structure
- Context-sensitivity
- Ability to plan actions at different levels of abstraction. In particular, ability to combine deliberative vs. reactive behavior.
- Strict optimality is often not necessary. In most real-time applications, it is unreasonable to run an algorithm that will take ten times more trials to learn better performance in order to gain only one-twentieth increase in that performance. This is especially true in nonstationary environments, where overconvergence can lead to overfitting.

1.3 Thesis Contributions

The goal of this thesis is to find a planning approach for autonomous vehicles that possesses the desired characteristics outlined in Section 1.2, to design an appropriate architecture implementing that approach, and to find parameters that would make it applicable to real data.

The contributions of this thesis are:

- an analysis of possible planning approaches drawn from existing literature (Chapter 2)
- development of a planning approach for autonomous vehicles combining desired features from possible approaches (Section 1.4)
- detailed analysis of reinforcement learning techniques and their combination with function approximation systems, confirming the choice of the proposed planning approach (Chapter 4)

- design of an architecture of specific algorithms implementing the chosen planning approach on real data (Chapter 5)
- formulation of a survey mission for autonomous helicopters as a reinforcement learning problem, which, unlike traditional formulations, permitted planning without knowledge of future states of the environment. (Chapter 3)
- The proposed context-sensitive planning approach for autonomous vehicles was tested on the survey mission example. It was shown to outperform both the more traditional global planning approach and the linear context-switching approach in low-noise and high-noise environments.

1.4 Overview of the Proposed Planning Approach

The diagram below outlines the components of the proposed planning approach.

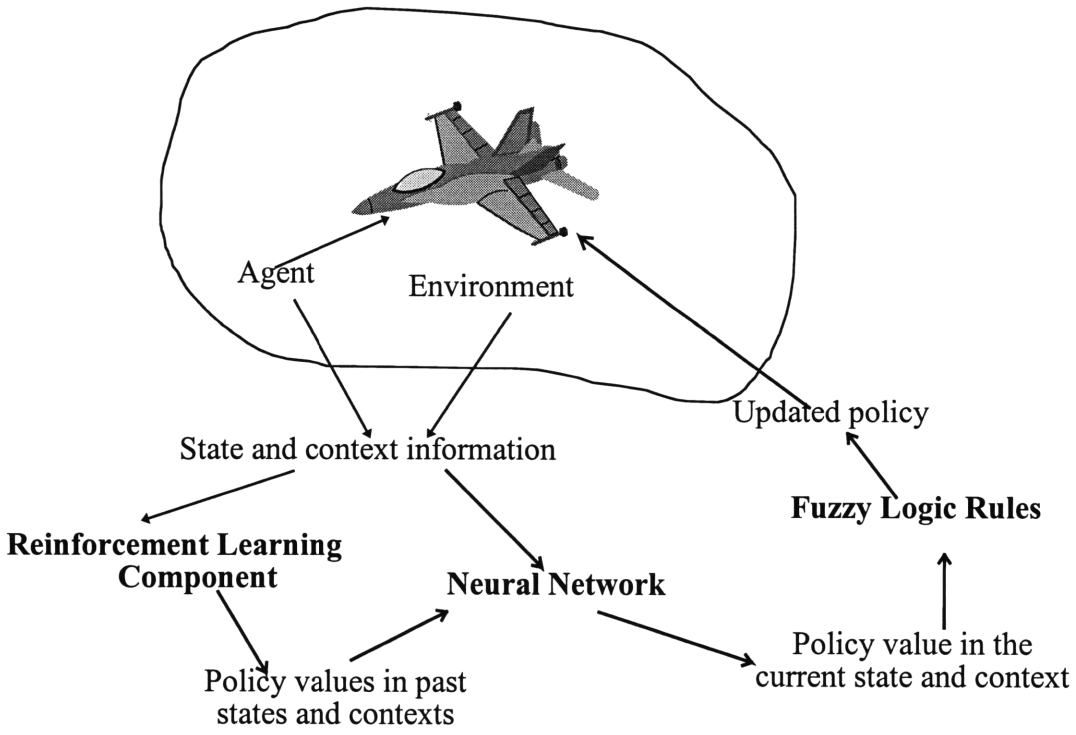


Figure 1.

Policy evaluation is the main aspect of the proposed planning approach. In many cases, policy updating can be expressed as simply choosing the policy with the highest value out of several generated alternatives. An adequate scheme for policy evaluation implies safer operation of the vehicle by allowing it to update its policies in a timely manner. At the same time, such a mechanism improves vehicle operation efficiency by not spending resources on updating a policy that seems to be performing poorly now but which is expected to get higher rewards in the future. For example, assigning a low value to the policy maintaining high speed when the operating context changes clear to obstructed will signal to the vehicle the need to slow down. Similarly, a slow motion can be well justified in a clear operating field if the context is expected to change to the obstructed operating field.

The theory of reinforcement learning provides an approach for policy evaluation based on the expected sum of future rewards obtained during plan execution. As will be shown in Chapter 3, the combination of reinforcement learning and function approximation architectures addresses the following issues mentioned in the previous section:

1. continuous policy evaluation
2. not requiring knowledge of the future environment states
3. working in continuous state and action spaces
4. working with noisy, nonlinear, nonstationary and unknown state transition function
5. accounting for evolving goal structure
6. working with suboptimal policies and improving them in the process.

The remaining issues are addressed by the following two features of the proposed planning architecture.

As was indicated in Section 1.1, a policy's value depends on the level of abstraction at which the most pressing goals are defined. At certain times, a more strategic behavior for achieving long-term goals is more beneficial, while at other

times a more tactical behavior for dealing with local environment might be necessary. For example, during favorable weather conditions an autonomous helicopter performing a survey mission might be able to fly faster to cover more search area. During unfavorable weather conditions with poor observability, on the other hand, the helicopter must fly slower to reduce the chance of colliding with poles and other aircraft. The proper level of abstraction often depends on the context present in the environment, which might be difficult to determine due to sensor noise and partial observability. The proposed planning architecture will use a special neural network configuration for context-sensitive policy evaluation.

The policy updating capability in the presented planning approach will be provided by fuzzy logic rules. These rules will process information provided by the other two architectural components about the policy value and the environment context. In addition, they can use information about mission-dependent constraints affecting vehicle's behavior, such as constraints on how fast state variables such as speed or altitude can be changed. The transparency of the rule-based approach will allow easy monitoring and adjustment of the policy updating mechanism in the vehicle. Also, it will allow the field experts to encode relevant information about known features of operating environment and mission characteristics in addition to the information gathered during mission executions.

Chapter 2

Analysis of Existing Planning Approaches

Other researchers have considered different aspects of the general planning problem, exploring particular techniques or formalisms designed to work well within specific areas, such as graph searching or task scheduling. Little attention, however, has been directed towards finding approaches that are appropriate for the entire problem as was described in Section 1.1. An overview of existing planning approaches is presented below, and their appropriateness for the planning problem as described in Chapter 1 will be analyzed. The approaches are organized in the following categories:

- classical operations research planning
- reactive planning
- planning using reinforcement learning
- extensions of reinforcement learning
- context-sensitive decision making

Some of the cited works may fall in more than one category, as will be pointed out in passing.

2.1 Classical Operations Research Planning

Most of the existing work in planning considers situations in which possible future environment states are known. The planning problem then becomes an operations research problem of finding the best path through a graph of possible alternatives. This approach has been used in designing the autonomous submarine at the C. S. Draper Laboratory [Ricard, 1994]. The A* search has been used to find the best path following a terrain with *a priori* known features. The same approach has been used in the design of an autonomous submarine at the Texas A&M University [Carroll et. al., 1992]. Alternatively, a global search technique such as simulated annealing can be used to generate possible vehicle trajectories and then select the one

with the highest score [Adams, 1997]. However, the cost function in such approaches evaluates the trajectory with respect to the *known features* of the future environment.

Extensions of classical planning work, such as CNLP [Peot and Smith, 1992] and CASSANDRA [Pryor and Collins, 1996] have considered nondeterministic environments. For each action in the plan, there is a set of possible next states that could occur. A drawback of these approaches is that they do not use information about the relative likelihood of the possible states. More advanced planning algorithms such as BURIDAN [Kushmerick, et. al., 1995] and C-BURIDAN [Draper et. al., 1993] operate in environments with probability distributions over initial states and state transitions. However, they assume discrete state spaces and fixed state transition probabilities. Therefore, these approaches are not applicable to noisy continuous state spaces with nonstationary dynamics.

2.2 *Reactive Planning*

Reactive planning approaches constitute roughly the other side of the spectrum of planning methods. One type of reactive approach is that of case-based learning. Some recent papers providing a good overview of this field are [Aha, 1997], [Aamodt, 1993], [Murdock et. al., 1996], and [Bergmann et. al., 1995]. Instead of relying solely on the knowledge of the problem domain, case-based planning utilizes the specific knowledge of previously experienced, concrete problem situations (cases). A new plan is found by finding a similar past case, adapting it to the problem at hand, and using the obtained result as the new plan. One of the main trade-offs in case-based planning is that of balancing the time spent on retrieving a more similar solution with the time spent on adapting a less similar solution. Case-based planning and case-based reasoning (CBR) in general has the advantage of not requiring knowledge about exact future environment states. The main drawback of case-based reasoning is its inability to deal explicitly with noisy nonstationary data, since there is no mechanism in CBR for deciding whether a case should be stored or discarded as

noise. To solve this problem, CBR can be extended by storing abstracted prototypical cases and constantly revising the case base to maintain only relevant information.

The above approach for extending CBR is best formalized by treating cases as fuzzy logic rules [Plaza et. al., 1997, Dutta and Bonissone, 1993]. Recent advances in data-driven fuzzy reasoning allow for an automatic determination of the applicability of each rule [Lin and Lee, 1996, Tsoukalas and Uhrig, 1997]. This has led to a surge of interest in designing fuzzy logic controllers for autonomous systems [Baxter and Bumby, 1993, Bonarini and Basso, 1994, Voudouris et. al., 1994]. The advantage of using automatic fuzzy logic rules to guide the behavior of autonomous vehicles is that they allow the vehicle to maintain its safety in complex and unpredictable environments. However, this reactive approach cannot reason about the temporal consequences of agent's actions for achieving its long-term high-level goals. For example, a fixed set of rules specifying obstacle avoidance maneuvers does not tell the vehicle in which direction it should steer during such maneuvers. This information can only be obtained by evaluating the sequence of future situations that are likely to arise as a result of a certain maneuver.

As will be shown in chapter 4, the theory of reinforcement learning provides methods for evaluating such sequences. The planning approach presented in this thesis combines the reactive fuzzy rule-based approach for recommending tactically correct actions with reinforcement learning for evaluating long-term strategic effects of these actions.

2.3 Planning Using Reinforcement Learning

The theory of reinforcement learning encompasses a set of planning techniques based on the Bellman's optimality principle and dynamic programming [Bellman, 1957]. As opposed to classical planning algorithms that perform an exhaustive search in the space of possible plans, the dynamic programming algorithm constructs an optimal plan by solving state recurrence relations (see section 4.2 for more details).

The dynamic programming algorithm suffers from the same problem as the classical graph-based search techniques: they all require knowledge of future environment states. The theory of reinforcement provides algorithms for incrementally estimating the state values as the states are visited over and over again rather than by working backward from a known final state.

Reinforcement learning can also be viewed as an extension of case-based reasoning, where the case (action) that would lead to a future state with the highest value is chosen at each decision moment. In other words, the theory of reinforcement learning provides algorithms for learning the similarity metric for case-based action selection.

The above features of reinforcement learning made it an attractive approach for determining optimal policies for autonomous vehicles [Asada et. al., 1994, Gachet et. al., 1994]. However, there are two main problems in reinforcement learning that limit its real-world applications: the curse of dimensionality and partial observability. The first problem refers to the fact that when the size of the state-space grows the chance of encountering the same state twice decreases. Therefore, the agent will have to choose its actions randomly almost at every time instance, as its state at that instance is likely to be new and unexplored. The second problem refers to the fact that the exact state of the vehicle can be very difficult to determine in noisy environments with poor sensor data.

2.4 Extensions of Reinforcement Learning

The standard approach to address the curse of dimensionality in reinforcement learning is to estimate values of new states using function approximation systems such as neural networks [Patek and Bertsekas, 1996], fuzzy logic [Berenji, 1996], or local memory-based methods such as generalizations of nearest neighbor methods [Sutton et. al., 1996]. Instead of waiting until all states will be visited several times in order to determine the best action in each state, the function approximation approach allows to assign similar values to similar states and

consequently to take similar actions in them. This is essential for agents operating in continuous state and action spaces.

Combinations of reinforcement learning and function approximation systems have led to a remarkable success in domains such as game playing [Tesauro, 1992] and robotics [Moore 1990, Lin, 1992]. For example, Tesauro reports a backgammon computer program that has reached grand-master level of play, which has been constructed by combining reinforcement learning with neural networks. However, the above works explored situations in which there was no unobservable hidden structure in the agent's state space.

Some approaches have been proposed for addressing the issues of hidden states in reinforcement learning [Parr and Russel, 1995, Littman et. al., 1995, Kaelbling et. al., 1997]. A mobile robot Xavier has been designed at Carnegie Mellon University to operate in an office environment where corridors, foyers and rooms might look alike [Koenig and Simmons, 1996]. This research is based on the theory of Partially Observable Markov Decision Processes (POMPD) [Smallwood and Sondik, 1973]. The main limitation of this approach is that it becomes computationally intractable for even moderate sizes of agent's state space [Littman et. al., 1995], which prevents its application to continuous state spaces. In addition, it doesn't account for possibilities of a hierarchical hidden structure, such as presence of hidden contexts that influence all states that they cover.

2.5 Context-Sensitive Decision Making

The problem of hidden contexts is very important in planning for autonomous vehicles. The operating environment for such vehicles is rarely uniform, and appropriate types of behavior must be chosen for each context. For example, navigation and survey behavior of an autonomous helicopter should differ in good and bad weather conditions, in urban and rural areas, in open fields and mountain ranges. The exact ways in which these contexts affect the outcome of vehicle's actions

can be hidden due to limited number of sensors and limited background information about the environment.

Recent research in the AI community has begun to address explicitly the problem of hidden contexts in the agent's environment [Silvia et. al., 1996, Turney, 1997, Widmer and Kubat, 1996]. The main limitation of these works is that they have been concentrating on discrete symbolic domains, which prevents their use for control of continuous autonomous vehicle variables such as speed or altitude.

The problem of hidden contexts in continuous variables has been considered in time series forecasting [Park and Hu, 1996, Pawelzik et. al., 1996, Teow and Tan, 1995, Waterhouse et. al., 1996, Shi and Weigend, 1997]. This has led to the development of the so-called "mixture of experts" models, in which each expert is represented by a function approximation architecture. Each expert specializes in forecasting time patterns in its own context, and their forecasts are weighed by the probabilities of each context being present in the data. The weights can either be determined by a supervisory gating network or by the experts themselves.

The neural network component in this work is based on the gated experts architecture presented in [Weigend and Mangeas, 1995]. The above architecture has been designed to recognize contexts that are readily apparent in the input data. For example, one of the data sets on which the above architecture was tested consisted of regimes of chaotic inputs and noisy autoregressive inputs.

The gated experts architecture in this thesis extends the above architecture by allowing the gate to consider both the raw input data and past errors of the experts. This extension makes the gated experts architecture applicable to domains where data targets are not future values of inputs, as in time series modeling, but are based on a separate reward function, as in reinforcement learning. This thesis demonstrates the use of the gated experts architecture for predicting long-term policy values rather then just immediate future rewards.

Chapter 3

Survey Mission Example

3.1 Survey Mission Overview

The planning approach presented in this thesis can be applied to any autonomous vehicle. For concreteness, the usage will be demonstrated on the case autonomous aerial vehicles (AAV's). The mission of aerial survey for autonomous helicopters will be used as an example for testing the proposed planning framework. The proposed framework for context-sensitive policy evaluation and updating will be applied to the policy for controlling cruising speed and altitude of a helicopter performing a visual survey. In this case, speed and altitude are critical parameters, and their proper control is a very difficult task. On the one hand, speed and altitude control the resolution and hence the quality of collected information. On the other hand, speed and altitude are the main parameters of navigation, which is highly dependent on environmental conditions such as weather and obstacles present. Hence, any planner has to balance the values of speed and altitude to achieve the strategic goal of information gathering with the values that are required for the tactical goal of navigation and vehicle safety.

In visual survey missions, the resolution at which information is collected determines abstraction level of the information. If resolution is too high, some objects might be very hard to piece together out of their parts. If resolution is too low, not enough details will be present to distinguish objects from each other. Examples of this tradeoff include the tasks of recognizing a car as well as reading its license plate, creating an informative map of a village that identifies houses as well as possible weapons, determining a path to the goal for a submarine as well as finding mines or obstacles on the path, etc. The proposed policy evaluation framework will allow one to solve different problems of this type.

The policy goal is simply to collect pictures carrying the largest amount of information according to the Shannon's measure of information content [Shannon,

1948]. As is conventional for the task of classifying visual images, this measure is defined as $\Sigma\, p_i \log p_i$, where p_i is the probability of the image belonging to class i. The measure is maximized when one of the p_i's is equal to 1 and the rest are 0's, which corresponds to the total confidence in classification. The measure is minimized when all p_i's are equal, which corresponds to a totally uninformative set of pictures.

Information content is critically dependent on the resolution at which the pictures are taken. In the mission of aerial survey, higher speed and altitude will lead to a lower resolution, and vice versa. The *value* of the policy is the expected amount of information obtained from future pictures if speed and altitude continue to evolve with the same dynamics. However, this dynamics can be modeled only approximately because of sensor uncertainty, partial observability, and the need for unforeseen obstacle avoidance maneuvers. I will assume that the policy is determined by the desired values that it sets for speed and altitude and to which these variables are set by the navigation controller after unforeseen disturbances.

The proper values for desired speed and altitude depend on the context present in the environment. For simplicity, the environment will be assumed to contain just two contexts corresponding to favorable and unfavorable conditions. In favorable contexts with good visibility, pictures at a low resolution encompass more area and at the same time contain enough detailed information for proper image classification. In unfavorable contexts when visibility is low due to dust, rain, or dusk, pictures need to be taken at a higher resolution to obtain better clarity of details. However, speed and altitude deviations from the desired values in each context will lead to a lower picture information content.

The context interpretation given above can be clearly be generalized to many different missions. For example, favorable contexts can represent situations in which giving more weight to behavior for achieving higher-level goals produces better results. Similarly, unfavorable contexts can represent situations in which giving more weight to reactive behavior for avoiding environment disturbances produces better results. Furthermore, the proposed framework can work with any number of hidden

contexts, which can represent other issues besides the level of abstraction required of actions. However, the chosen interpretation is an adequate paradigm for testing all ideas discussed in Section 1 as its solution will require addressing major issues of operating in complex, uncertain, and nonstationary environments.

3.2 *Survey Mission Specifications*

Helicopter's altitude and speed will be used as state variables for the information collection task. I will assume that because of vehicle's computational constraints, the state will be estimated at discrete time intervals.

In the proposed case study, I assume that under the existing path planning policy both speed and altitude evolve according to a nonlinearly transformed noisy autoregressive function

$$x_i(t+1) = \tanh[x_i(t) + \varepsilon_i], \quad i = 1, 2. \tag{1}$$

In the above equation, $x_1(t)$, $x_2(t) \in \mathbf{R}$ are correspondingly the current altitude and speed, and $\varepsilon_i \sim N(0, \sigma)$. The above equation models a nonlinear noisy autoregressive state transition function. The autoregressive nature of this function reflects the assumption that the speed and altitude of the helicopter cannot change abruptly, but depend on previous values. The last term in equation (1) is a noise term, which models uncertainty about future values of state variables due to unpredictable environment conditions, such as weather effects or obstacles that need to be avoided. The tanh transformation in the above equation reflects the assumption that both speed and altitude are constrained to lie within certain limits. These limits are scaled using tanh to the [-1,1] range, which corresponds to measuring the deviation of these quantities from a certain average.

The purpose of the proposed architecture will be to evaluate the path planning policy at each time t = T, i.e. to determine the expected sum of future rewards (t = T+1, T+2, ...) from following the policy starting at state x(T). The reward

function mapping states to rewards depends on the context present in the environment.

In favorable contexts, the reward is determined as follows:

$$r_1(x(t)) = 4*\exp(-k(u - d_f)) / (1+\exp(-k(u - d_f)))^2, \quad (2)$$

where $u = x_1(t) + x_2(t) + \varepsilon$, d_f is the optimal value for u in favorable contexts, and $\varepsilon \sim N(0, \sigma)$. In unfavorable contexts the reward is determined as follows:

$$r_2(x(t)) = 4*\exp(-k(u - d_u)) / (1+\exp(-k(u - d_u)))^2, \quad (3)$$

where d_u is the optimal value for u in unfavorable contexts. The reward function given above is motivated by the fact that when the speed or altitude is obviously too much out of range, the reward should be zero. At the same time, reward cannot grow arbitrarily since the most appropriate state values lie in between two bounds of being too high or too low. Therefore, the reward function was chosen to be the derivative of the sigmoid function, which is 0 for very small and large inputs and 1 at d_f in favorable contexts and at d_u in unfavorable contexts. The constant k is the scaling factor that controls how fast the reward rises from almost 0 to 1 and falls back.

The probability of a context starting to change at any given time is 1/N. The final reward function $r(t)$ is given by:

$$r(x(t)) = a*r_r(x(t)) + (1-a)*r_s(x(t)), \quad (4)$$

where the subscript s corresponds to the current regime, the subscript r corresponds to the previous regime, $a = \exp(-n\mathrm{T})$, and T is the number of time steps since the previous regime started changing. This functional form for the final reward function models a smooth change between the regimes. This corresponds to the assumption

that abrupt context changes rarely take place in the real world, and if they do, they cannot be determined with certainty.

As we will show, the proposed framework addresses two major issues in the problem formulation described above. First, it allows one to accurately evaluate the existing policy despite the presence of hidden contexts as well as general environment uncertainty, complexity, and nonstationarity. Second, as described in Section 1.4, such an accurate evaluation will allow the vehicle to change its policy to a different one with a higher value, if such exists. The capability to appropriately change the policy for controlling speed and altitude increases the efficiency of the vehicle's operation. Moreover, this capability enables the vehicle to operate in environments where no single course of behavior can be followed at all times. Tests confirming the above two capabilities of the proposed architecture will be presented in Chapter 6.

Chapter 4

Background for the Proposed Architecture

4.1 Optimal Control Formulation

The policy evaluation approach presented in this thesis has its roots in the theory of optimal control. This formulation is very general and is capable of representing a very large class of problems, including the classical planning problems. In terms of control theory, classical planning algorithms produce an open-loop control policy from a system model; the policy is applicable for the given initial state. Furthermore, classical planning is an off-line design procedure because it is completed before the system begins operating, i.e., the planning phase strictly precedes the execution phase. A closed-loop control policy is simply a rule specifying each action as a function of current, and possibly past, information about the behavior of the controlled system.

The central concept in control theory formulation is that of a state of the environment. It represents a summary of the environment's past that contains all relevant information to the environment's future behavior. The state transition function specifies how the state changes as a function of time and the actions taken by the agent. Each action taken in some state leads to a certain reward or punishment. The agent is trying to maximize the total reward obtained from choosing actions in all the states encountered. For example, the finite horizon discrete time optimal control problem is:

Find an action sequence $\boldsymbol{u} = (u_0, u_1, \ldots, u_{T-1})$ that maximizes the reward function

$$J(\boldsymbol{u}) = g_T(x_T) + \sum_{i=0}^{T-1} g_i(u_i, x_i)$$

subject to the state transition function

$$x_{i+1} = f_i(x_i, u_i, w_i), \qquad i = 0,\ldots,T\text{-}1,$$

$$x_0 : \text{ given}$$

$$x_i \in R^n, \qquad i = 1,\ldots,T$$

$$u_i \in R^m, \qquad i = 0,\ldots,T\text{-}1,$$

where w_i is a vector of uncontrollable environment variables.

4.2 Methods for Solving the Optimal Control Problem

There are several approaches for solving the optimal control problem that depend on the degree of complexity, uncertainty, and nonstationarity present in the environment. In the simplest case, when the time dependent state transition functions f_i are known with certainty and are deterministic, then nonlinear programming can be used to find the optimal sequence of actions $\boldsymbol{u}$ with a corresponding sequence of the environment states $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$.

In a more complex case of probabilistic state transitions and a small set of possible states, dynamic programming can be used to obtain an optimal policy $u_i = \mu_i(x_i)$, that specifies optimal action to take in every state x_i. The dynamic programming (DP) algorithm exploits the fact that the cost function has a compositional structure, which allows it to find the optimal sequence of actions by solving recurrence relations on these sequences rather than by performing an exhaustive search in the space of all such sequences. Backing up state evaluations is the basic step of the DP algorithm for solving these recurrence relations. For example, if a certain action a in state i is known to lead to state j under the current policy, then the value of this policy $V(i)$ in state i is:

$$V(i) = r(i) + V(j),$$

where $r(i)$ is the immediate benefit of taking the action a. Then, starting from the final state, actions are chosen to maximize the sum of the immediate benefit in the current state and the long-term value of the next state.

The dynamic programming algorithm does not apply if the state transition function is not known. This problem is commonly referred to as the curse of modeling. Traditionally, the theory of adaptive control has been used for these situations. In the adaptive control formulation, although the dynamic model of the environment is not known in advance and must be estimated from data, the structure of the dynamic model is fixed, leaving the model estimation as a parameter estimation problem. This assumption permits deep, elegant and powerful mathematical analysis, but its applicability diminishes for more complex and uncertain environments.

4.3 Solution With Reinforcement Learning

The most general approach for solving the optimal control problem under high complexity and uncertainty is to formulate it as a reinforcement learning problem [Kaelbling et. al., 1996]. Reinforcement learning provides a framework for an agent to learn how to act in an unknown environment in which the results of the agent's actions are evaluated but the most desirable actions are not known. In addition, the environment can be dynamic and changes can occur not only as a result of exogenous forces but also as a result of agent's actions. For example, the environment of an autonomous helicopter will change if the weather changes or if the helicopter flies into a densely populated area with heavy air traffic.

The theory of reinforcement learning defines mission planning and execution in terms of a plan-act-sense-evaluate operating cycle. At the first stage of this cycle, an action plan with the highest value is chosen for accomplishing high-level goals. At the second stage, the designed plan is executed in a manner that satisfies the environment constraints. At the third stage, the changes to the environment as a result of plan execution are sensed by the agent. At the fourth stage, the value of the current plan is updated based on the newly received sensor input.

More formally, during each step of interacting with the environment, the agent receives as input some indication of the current state x_i and chooses and action

u_i. The action changes the state of the environment, and the value of this state transition is communicated to the agent through a scalar reinforcement signal r. The agent is trying to choose actions that tend to increase the long-run sum of values of the reinforcement signal. It can learn to do this over time by systematic trial and error, guided by a wide variety of algorithms that have been developed for this problem. The above formulation is much more general than that of optimal control. It doesn't require the state transition functions f_i and the reinforcement functions $g_i(u_i,x_i)$ to be of any particular form or even be known. The agent's environment is also not restricted in any way. Of course, the price paid for the above relaxation is that the optimal action policy becomes much more difficult or even impossible to find.

Reinforcement learning differs from the more widely studied problem of supervised learning in several ways. The most important difference is that there is no presentation of input/output pairs to the agent. Instead, after choosing an action the agent is told the immediate reward and the subsequent state, but is not told which action would have been in its best long-term interests. It is necessary for the agent to accumulate useful experience about the possible system of states, actions, transitions and rewards to act optimally. Another difference from supervised learning is that these systems do not learn while being used. It is more appropriate to call them learned systems rather than learning systems. Interest in reinforcement learning stems in large part from the desire to make systems that learn from autonomous interaction with their environments.

Reinforcement learning allows for a flexible combination of top-down goal-oriented and bottom-up reactive planning. Bottom-up planning is naturally present in reinforcement learning, since the generated action policies depend on the state of the environment at each moment of time. The degree to which the agent's behavior is strategic and goal-oriented can be controlled by providing rewards for subgoals at different levels of abstraction. When rewards are given only for highest level subgoals, more tactical decisions will be left up to the agent, and its behavior will

become less strategic and more reactive. Similarly, when rewards are given for very low level subgoals, the agent will have less freedom in making local environment-driven decisions and will be more directed towards achieving the specified subgoals regardless of the environment. The first approach is more preferable for environments with high dynamics, complexity, and uncertainty, while the second approach is more preferable for simple and static environments. Thus, reinforcement learning can be used as a generalization of traditional goal-oriented planning techniques to environments with stochastic or even unknown dynamics.

4.4 Policy Evaluation in Reinforcement Learning

All reinforcement learning algorithms are based on estimating the policy value in each state of the environment. Policy value is defined as an averaged or discounted sum of rewards that can be obtained by starting in the current state and following the given policy. As discussed in the introduction, continuous policy evaluation is central maintaining the proper balance of strategic vs. tactical behavior. In many situations, given a mechanism for evaluating future effects of actions, policy updating stage consists simply of choosing the policy with the highest value.

The task of policy evaluation faces several problems in dynamic, complex, and uncertain environments. All standard reinforcement learning algorithms assume that it is possible to enumerate the state and action spaces and store tables of values over them. The values in these tables are updated after each interaction with environment in a manner specific to each algorithm. Except in very small and simple environments, storing and updating tables of values means impractical memory and computational requirements. This problem arises in complex environments and is commonly referred to as the curse of dimensionality. It is present in its extreme form in continuous state spaces. This creates the need for compact representations for state and action spaces in terms of abstracted feature vectors. Therefore, the notion of a look-up table has to be expanded to a more general *value function,* representing a mapping from these feature vectors to state values. For example, the vector of sensor

variables received by an autonomous helicopter at a certain time instant can be used to determine the value of the helicopter's path planning policy.

Besides addressing memory and computational requirements, compact representations will allow for a more efficient use of learning experience. In a large smooth state space we generally expect similar states to have similar values and similar optimal actions. This is essential for continuous state and action spaces, where possible states and actions cannot be enumerated as they are uncountably infinite. The problem of learning state values in large spaces requires the use generalization techniques for transfer of knowledge between "similar" states and actions. These techniques would lead to a significant speed up of learning policies in complex environments, and make reinforcement learning algorithms better suited for real-time operations.

Another problem for learning state values in reinforcement learning results from environment uncertainty. It is often the case that observations received by the helicopter do not possess the Markov property, which requires the current observation to contain all the information necessary for compute the probability distribution of the next observation. Hence, these observations cannot be treated as states of the environment. This problem is sometimes referred to as the hidden state problem. This type of uncertainty is different from the probabilistic uncertainty about the future states, and it requires special mechanisms for handling it. The standard approach to solving this problem is to fit a Hidden Markov Model (HMM) to the sequence of agent's observations, actions, and rewards [Shi and Weigend, 1997]. This model is sometimes referred to as a Partially Observable Markov Decision Process (POMPD) model [Kaelbling, Littman, and Cassandra, 1997]. In this model, a set of hidden states is assumed with a Markovian transition function. Using this assumption, the probability of the next observation is estimated based on a fixed length sequence of past observations, action, and rewards.

However, a fixed state transition function is unrealistic in nonstationary real world environments. In order to avoid this difficulty, the idea of a fixed model of the

environment should be abandoned: the two steps of estimating the current state and then estimating its value should be collapsed into a single step of estimating the value of the current situation based on the history past observations, actions, and rewards. This estimation can be done using either a model-based function approximator such as a neural network or a case-based function approximator such as a nearest neighbor method or a fuzzy logic system. Both the parameters and the structure of the function approximation architecture can be updated on-line to reflect the drift in a nonstationary environment.

Some reinforcement learning techniques have been proven to converge for function approximation systems that are quadratic in weights [Bradke, 1993]. However, all convergence results are relevant only to environments with fixed dynamics. If dynamics of state transitions or reward assignments is changing over time, then convergence to the optimal policy on a given data set might even be an undesirable result. Such convergence would lead to overfitting of the system on the existing data and will result in poor generalization on new data.

Besides slow drift in the state evaluation function due to environment nonstationarity, more abrupt changes due to switches in underlying hidden regimes can occur. In traditional treatment of the hidden state problem in reinforcement learning, the hidden states can change at every time step and thus represent high-frequency changes. In some environments, extra complications can arise due to lower frequency changes that represent more fundamental changes in the environment. For example, helicopter's altitude might constitute a state, and air traffic in helicopter's surroundings might constitute a context, both of which might be hidden (difficult to determine) due to poor weather conditions. A lot of work has recently been devoted to modeling such changes in the field of Artificial Intelligence, where they are referred to as changes in the hidden context [Widmer and Kubat, Taylor and Nakhaeizadeh]. In addition, such models began to appear in time series modeling [Weigend and Mangeas, 1995].

The problems with value estimation described above will be addressed by combining the reinforcement learning component with a mixture of experts neural network architecture. The proposed neural architecture was designed to operate on real-valued data arriving from changing hidden contexts. It was also designed to be more robust to overfitting than global models that disregard context information.

The proposed RL/NN combination is very general and makes only the mildest assumptions about the data coming from the environment. The required data consists of real vectors describing vehicle's state over time and scalar rewards for actions taken in each state. The state observations do not even have to contain all relevant information for determining future states. Furthermore, the relationship between state vectors and costs of actions taken can be stochastic and nonlinear. To my knowledge, no other algorithms for policy evaluation in reinforcement learning have been implemented for such general data.

Chapter 5

The Proposed Architecture

5.1 Reinforcement Learning Component

The RL component of the proposed planning approach will provide targets for the neural network component. Several issues arise in calculating the network targets. The first one concerns targets $d(t)$ for $t < T$. The available and relevant information for computing the target $d(t)$ consists of targets $d(\tau)$ for $t < \tau \leq T$ and outputs $y(\tau)$, as well as rewards $r(\tau)$ for $t \leq \tau \leq T$. The standard dynamic programming algorithm of value iteration is

$$V(x(t)) = \max_{a}\left(E[r(x(t),a)] + \gamma \sum_{x(t+1)} P(x(t),x(t+1),a) \cdot V(x(t+1))\right),$$

where $E[r(x(t),a)]$ is the expected value of the reward received at state $x(t)$ for taking action a, $P(x(t),x(t+1),a)$ is the probability of transferring to state $x(t+1)$ after choosing action a in state $x(t)$, and γ is the discounting factor. However, the above algorithm is not applicable to the problem of interest for several reasons. First, the probability $P(x(t),x(t+1),a)$ does not exist because the state space for the problem of interest is continuous. Second, even if the state space could be discretized and averages over the data set would be used as probabilities, the value iteration algorithm could still not be used because the transition probability model for the system is assumed to be unknown.

The simplest and the most common solution to this situation is to use a sampling algorithm such as TD(0) [Kaelbling et. al., 1996]. In this algorithm, the value of a state $x(t)$ is updated after receiving the reward $r(t)$ and moving to state $x(t+1)$ according to:

$$V(x(t)) = y(t) + \alpha[r(t) + \gamma * V(x(t+1)) - y(t)],$$

where $y(t)$ is the forecast for $V(x(t))$ and α is a learning rate. This algorithm is analogous to the standard value iteration - the only difference is that the sample is drawn from the real world rather than from a simulation of a known model. The key idea is that $r(t) + \gamma * V(x(t+1))$ is a sample of the value of $V(x(t)$, and it is more likely to be correct because it incorporates the real reward $r(t)$.

The next issue that arises in computing the NN targets is what should be the value $V(x(t+1))$. The two possible choices are the network output $y(t+1)$ and the computed target (desired value) $d(t+1)$. The difference between the two alternatives is best illustrated on the example of the second to last data point. This task resembles the situation of on-line training, where we are trying to compute d(T-1) after transferring to state x(T). In this situation the desired value d(T) is unknown, and the network output y(T) has to be used instead. However, when we go back in time beyond the second to last data point, the choice $d(t+1)$ for $V(x(t+1))$ would be more informative than $y(t+1)$, since the later was generated without the knowledge of future rewards $r(\tau)$ for $\tau \geq t$.

Finally, the question of calculating the target d(T) for the last data point has to be resolved. TD(0) as well as all other reinforcement learning algorithms provide the target for a state value based on the expectation of actual value of the next state encountered. Since no actual state is available for the last data point, a naive expectation method has to be used:

$$V(x(T)) = E\left[\sum_{\tau=T}^{\infty} r(\tau)\gamma^{\tau}\right] = \frac{E[r(\tau)]}{1-\gamma}.$$

The equations for the evolution of speed and altitude given in chapter 3 were supposed to model a common situation in which speed and altitude oscillate around certain middle values. To aid NN training, the data in the experiments was normalized to zero mean and unit variance. Hence, $E[r(\tau)] = 0$.

If the learning rate α is slowly decreased and the policy is held fixed, TD(0) is guaranteed to converge to the optimal value function for finite state spaces in which every state is visited infinitely often. The above convergence result is not applicable to continuous state spaces. However, as was discussed in the introduction, the convergence issues do not arise in nonstationary environment, where overconvergence results in overfitting.

5.2 Neural Network Component: Gated Expert Overview

The policy evaluation function in this thesis is approximated by a neural network architecture. It is trained on the state-value pairs provided by the RL component to forecast the policy value (expected sum of future rewards) in each state. In addition to addressing the extreme curse of dimensionality present in continuous state spaces as described in chapter 4, this architecture will also address the problem of hidden contexts.

The NN component is be based on the Gated Experts (GE) architecture [Weigend and Mangeas, 1995].

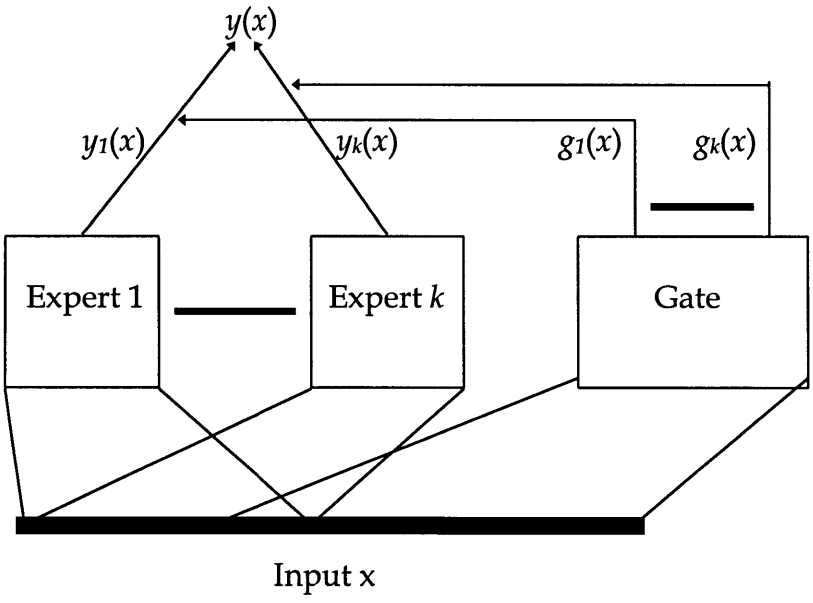


Figure 2.

The GE architecture consists of a set of experts and a gate, which can be arbitrary input-output models with adaptive parameters (neural networks in this thesis). When an input pattern is presented to the GE architecture, each expert gives its version of the output and the gate gives the probability of each of the expert's outputs being correct. The final GE output is obtained by adding the expert's individual outputs weighted by the corresponding probabilities.

The advantages of a mixture of experts architecture in uncertain environments can be formulated in more precise mathematical terms. In regression analysis, a committee forecast can be obtained by taking a weighted average of forecasts of N individual models. In that case, using any convex error function, the expected error of the committee models is less than or equal to the average error of N models acting separately. [Bishop, 1995] This result can easily be shown using the Cauchy's inequality for the sum squared error function:

$$\left(\sum_{i=1}^{N}\varepsilon_i\right)^2 \le N\sum_{i=1}^{N}\varepsilon_i^2 .$$

In fact, the performance of a committee can be shown to be better than the performance of any single model used in isolation [Bishop, 1995].

Also, the idea of committee averaging underlies Bayesian data modeling. The probability distribution of some quantity Q given a data set D and different models H_i can be written as

$$p(Q \mid D) = \sum_i p(Q, H_i \mid D) = \sum_i p(Q \mid D, H_i)\, p(H_i \mid D) .$$

Besides fighting uncertainty in the form of probabilistic noise, the mixture of experts architecture is capable of reducing risk in decision making. This was observed long ago in the fields of economics and finance. For example, in portfolio

management, the process of forming a committee of models is *called risk diversification*. In particular, it is well known that diversification in stock market investments can eliminate the firm-specific nonsystematic risk leaving only the omnipresent systematic risk associated with uncertainty in the market as a whole.

Finally, the gated experts architecture has definite advantages in nonstationarity environments [Weigend and Mangeas, 1995]. For example, a process that we want to model may change between contexts, some of which are similar to those encountered in the past. In this case, even though a single global model can in principle emulate any process, it is often hard to extract such a model from the available data. In particular, trying to learn contexts with different noise levels by a single model is a mismatch, since the model starts to extract features in some context that do not generalize well (local overfitting) before it has learned all it potentially could in another context (local underfitting). For such processes, an architecture capable of recognizing the contexts present in the data and weighing expert's opinions appropriately is highly advantageous. If, in addition, each expert will learn to specialize in making decisions in a separate context, then each of them will be solving a much simpler task than modeling the whole process. Another motivation for different experts in different contexts is that they can individually focus on a subset of input variables that is most relevant to their specific context.

5.3 Gated Experts Details

The following are the main implementational features of the gated experts architecture as described in [Weigend and Mangeas, 1995]. The targets for the GE architecture are assumed to be normally distributed with a certain input-dependent mean and a context-dependent variance. Each expert gives its forecast for the target mean, and assigns the same variance to all its forecasts. During the training process, each expert tries to determine the context in which its forecasts have the least error and adjust its variance according to the noise level in that context. The architecture is trained using the Expectation-Maximization (EM) algorithm. In this algorithm, the

gate is learning the correct expert probabilities for each input pattern, while each expert is simultaneously learning to do best on the patterns for which it is most responsible. Only one expert is assumed to be responsible for each time pattern, and the gate's weights given to each expert's forecast are interpreted as probabilities that the expert is fully responsible for that time pattern.

The goal of the training process is to maximize the model likelihood on all the time patterns. The likelihood function is given by

$$L = \prod_{t=1}^{N} P(y(t) = d(t)|x(t)) = \prod_{t=1}^{N}\sum_{j=1}^{K} g_j(x(t),\theta_g) P(d(t)|x(t),\theta_j)$$
$$= \prod_{t=1}^{N}\sum_{j=1}^{K} g_j(x(t),\theta_g) \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(d(t)-y_j(x(t),\theta_j))^2}{2\sigma_j^2}\right),$$

where N is the total number of training patterns, K is the total number of experts in the architecture, $y(t)$ is the output of the whole GE architecture, $d(t)$ and $x(t)$ are the target and the input for pattern at time t, $P(y(t) = d(t)|x(t),\theta_j) = P(d(t)|x(t),\theta_j)$ is the probability that $y(t) = d(t)$ given that expert j is responsible for the time pattern at time t, and $g_j(x(t),\theta_g)$ is the probability that the gate assigns to the forecast of expert j. In addition, θ_j and θ_g are parameters for expert j and the gate, which are optimized during training.

The actual cost function for the GE architecture is the negative logarithm of the likelihood function:

$$C = -\ln L = \sum_{t=1}^{N} -\ln\left[\sum_{j=1}^{K} g_j(x(t),\theta_g) \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(d(t)-y_j(x(t),\theta_j))^2}{2\sigma_j^2}\right)\right].$$

The EM algorithm consists of two steps: the E-step and the M-step. During the E-step, the gate's targets are computed:

$$h_j(t) = \frac{g_j(x(t),\theta_g)P(d(t)|x(t),\theta_j)}{\sum_{k=1}^{K} g_k(x(t),\theta_g)P(d(t)|x(t),\theta_k)}.$$

During the M-step, the model parameters are updated as follows:

$$\sigma_j^2 = \frac{\sum_{t=1}^{N} h_j(t)(d(t) - y_j(x(t),\theta_j))^2}{\sum_{t=1}^{N} h_j(t)}.$$

The variance of the *j*th expert is the weighted average of the squared errors of that expert. The weight is given by $h_j(t)$, the posterior probability that expert j is responsible for pattern t. The denominator normalizes the given weights.

Since the cost function essentially minimizes the squared errors over all time patterns, the derivative of that function with respect to the output of each network is a scaled linear function. Thus, the weight changes in the expert networks are proportional to the difference between the desired value $d(t)$ and the expert output $y_j(t)$:

$$\frac{\partial C(t)}{\partial y_j(t)} = -h_j(t)\frac{1}{\sigma_j^2}(d(t) - y_j(x(t),\theta_j)).$$

The above learning rule adjusts the expert's parameters such that the output $y_j(t)$ moves toward the desired value $d(t)$. However, two factors specific to the GE architecture modulate the standard neural network learning rule. The first factor $h_j(t)$ punishes the experts for their errors according to their responsibility for the given time pattern. In other words, the expert whose context is more likely to be present at time t is trying harder to learn that time pattern. This allows for experts'

specialization in *contexts* rather than training one global model. The second factor, $\frac{1}{\sigma_j^2}$, punishes more for the same error and the same $h_j(t)$ the expert whose forecast precision was thought to be smaller. This allows the experts to adjust their variances to fit exactly the noise level present in their contexts.

The final outputs of the gating network g_j represent probabilities that the present context is best suited for each expert. Therefore, the raw outputs of the gate s_j have to be scaled to add up to 1, which is done using the conventional *softmax* function:

$$g_j = \frac{\exp(s_j)}{\sum_{k=1}^{K} \exp(s_k)}.$$

The *softmax* function enables "soft" competition between experts in which the output of each expert has some weight instead of giving the weight of 1 to the most likely expert. Such soft competition is desirable in high-noise environments, where no expert can in reality be totally responsible for each time pattern.

The weight changes in the gating network are proportional to the following quantity:

$$\frac{\partial C(t)}{\partial s_j} = -(h_j(t) - g_j(x(t), \theta_g)).$$

In the above equation, $h_j(t)$ is the posterior probability of expert j having the best fit for the given context - it is calculated using both input and target information. g_j, on the other hand, is calculated using only the input information. Thus, the above learning rule results in gate's outputs, which give predicted fitness of each expert to the current context, to approximate the actual expert's fitness to the context.

5.4 Fuzzy Logic Component

Fuzzy logic and neural networks are complementary technologies in the design of intelligent systems. Each method has merits and drawbacks. Neural networks are essentially low-level computational structures and algorithms that offer good performance dealing with sensory data. Neural networks possess a connectionist structure with fault tolerance and distributed representation properties that result in good learning and generalization abilities. However, because the internal layers of neural networks are always opaque to the user, the mapping rules in the network are not visible and difficult to understand. Furthermore, the convergence of learning is not guaranteed. Fuzzy logic systems, on the other hand, provide a structured framework with high-level fuzzy IF-THEN rules. An example of such a rule is: IF (the car ahead in your lane is breaking FAST) AND (the car behind you in the lane is FAR) THEN (turn left SHARP). The transparency of the rule-based approach allows human experts to incorporate a priori domain knowledge into such a system, since a large part of human reasoning consists of rules such as given above. However, since fuzzy systems do not have inherent learning capability, it is difficult for a human operator to tune the fuzzy rules and membership functions from the training data [Lin and Lee, 1996].

The above discussion suggests a natural combination of fuzzy logic and neural networks within the proposed framework. The learning qualities of neural networks make them a good fit for continuously evaluating policies in nonstationary environments. At the same time, the transparency of fuzzy logic rules is advantageous for policy updating mechanisms, for which human monitoring is desirable.

For the problem formulated in chapter 3, the following fuzzy rule structure was used for policy updating. The rule antecedents consisted of the following variables: policy value (LOW, or HIGH) and speed + altitude (LOW, or HIGH). The consequent was the desired value for speed + altitude (SMALL or LARGE). Ideally,

the desired value should be distributed among speed and altitude according to the current environment constraints on these variables. For simplicity, the desired value was evenly distributed in the current version of the system.

The following two fuzzy rules were used:

1. IF (policy value estimate is SMALL) AND (speed + altitude is SMALL) THEN (next desired value for speed + altitude is LARGE).
2. IF (policy value estimate is SMALL) AND (speed + altitude is LARGE) THEN (next desired value for speed + altitude is SMALL).

The membership function representing SMALL was $0.5\tanh(-kx)+0.5$, and representing LARGE was $0.5\tanh(kx)+0.5$. The antecedents generated by equations in chapter 3 were empirically determined to fall most often in the range of [-1,1]. Therefore, the scaling constant k was chosen to be 2 in order to capture most of the nonlinearity of the tanh function over the range of [-1,1]. The membership function for SMALL consequent was

$$\begin{cases} 0.5-0.5x, & \text{for } x \in [-1,1] \\ 0 & \text{otherwise} \end{cases}$$

For LARGE consequent, the membership function was

$$\begin{cases} 0.5+0.5x, & \text{for } x \in [-1,1] \\ 0 & \text{otherwise} \end{cases}$$

The AND operator was implemented using multiplication of the membership grades. The final desired value was obtained using the centroid defuzzification procedure:

$$DesiredVal = \int_{-1}^{1}[x(0.5+0.5x)s_1 + x(0.5-0.5x)s_2] = \frac{1}{3}(s_1 - s_2),$$

where s_1 is the strength of rule 1, and s_2 is the strength of rule 2. The strength of each rule was found by applying the membership functions to each of the antecedent variables and multiplying the results as prescribed by the AND operator. The simple interpretation of the above computational procedure is as follows: if a greater need is present for the desired value to be large, as indicated by a stronger result of rule 1,

then $s_1 - s_2 > 0$ and the desired value is greater than 0. The opposite effect holds when there is a greater need for the desired value to be small. The fuzzy rule extension of this reasoning allows to find *exactly* how large or small the desired value should be.

Chapter 6

Results and Discussion

6.1 Experimental Setup

The proposed planning architecture was tested on the data generated by equations given in Section 3.2. The tests were designed to demonstrate the two most important capabilities of the proposed architecture: 1. accurate policy evaluation despite changes in the hidden contexts 2. the possibility of effective policy updating.

The first capability was tested using the cross-validation approach. This is the standard approach for testing the results of a trained model having only a limited amount of noisy data. The model was trained on a set of data called the training set. After each training epoch, the model was applied to another set of data called the validation set, and the model error on the validation set was recorded. The training continued until either the training cost stopped changing or the validation cost began to increase. The first event implies that model parameters have come to a vicinity of a plateau or a local minimum. In this case, training was stopped to simulate real-world time limitations. The second event implies that the model started overfitting the training data, i.e. started fitting itself to the noise. In this case, training was stopped to preserve the model generalization abilities. After stopping the training, the model was applied to a third data set called the test set, and its cost was measured. The model cost on the test set is a point sample of the cost which the model is expected to incur on the new data. In order to obtain a more representative performance index, the test set was regenerated many times, and the average test cost was measured. By increasing the number of test sets, this final cost can be made to approximate arbitrarily closely the true cost that the model is expected to incur on the new data.

The following procedure was used to choose parameters for the experiments. To find the appropriate reward scaling parameter k in the reward equation, $E[r(\tau)]=0.5$ would have to be solved for k. However, this equation did not seem to be solvable analytically, and an iterative procedure was used to find k that resulted in a

value of $E[r(\tau)]$ to be reasonably close to 0.5. From this, the value k=3.52 was chosen, for which $E[r(\tau)]$=0.501.

The decay constant n in the final reward equation was chosen to be 1. The optimal sum d_f of scaled speed and altitude deviations in favorable contexts was 0.5, and d_u corresponding to unfavorable contexts was -0.5. The contexts were switched according to a uniform distribution, switching on average every 50 steps. 500 data points were used for training and 500 for testing. A smaller set resulted in overfitting that was difficult to manage, while a larger set required too much training time without an adequate gain in performance.

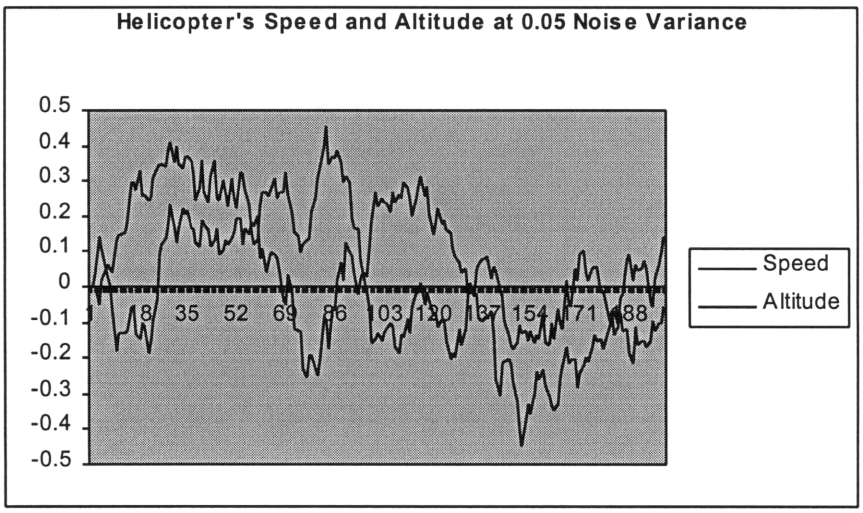
The learning rate of the TD(0) algorithm was fixed at 1 to simplify interpretation of the model cost. A smaller learning rate would result in targets being dependent on the network outputs. This would eliminate the benchmark of the error of 0 forecast, against which the model performance could be judged.

There were two experts in the model, each with one hidden layer of 10 sigmoid units. The gate had one layer of 10 sigmoid units. Both experts and the gate were trained using a second order Quasi-Newton method BFGS, as described in [Press et. al., 1992]. At time t, each expert had as inputs $x_1(t) + x_2(t)$, $x_1(t - 1) + x_2(t - 1)$, and $r(t - 1)$. The gate had as inputs exponentially smoothed expert's errors from past time patterns.

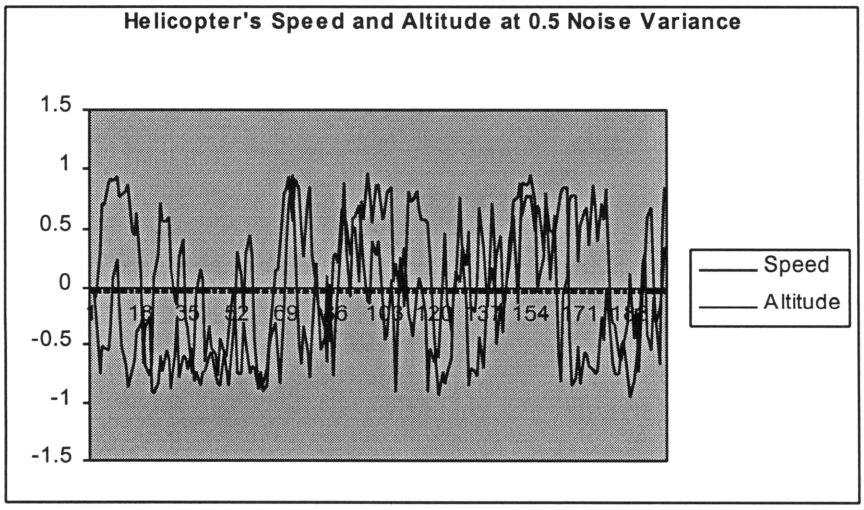
The expert's and gate's parameters θ_j and θ_g were randomly chosen at the beginning of each run. The network layers were optimized separately, as I found that they required very different step sizes in parameters. A weight decay factor λ= 0.1 was used for each layer to reduce overfitting: a penalty term of $\lambda \|W\|$ was added to the cost function for each layer, where $\|W\|$ is the Euclidean norm of the vector of weights in the corresponding layer.

6.2 Results and Discussion

Graphs of helicopter's speed and altitude for two different noise levels are shown below. Two hundred data points were used to make these graphs more readable. The data was generated using equation 1 in chapter 3.



Graph 1

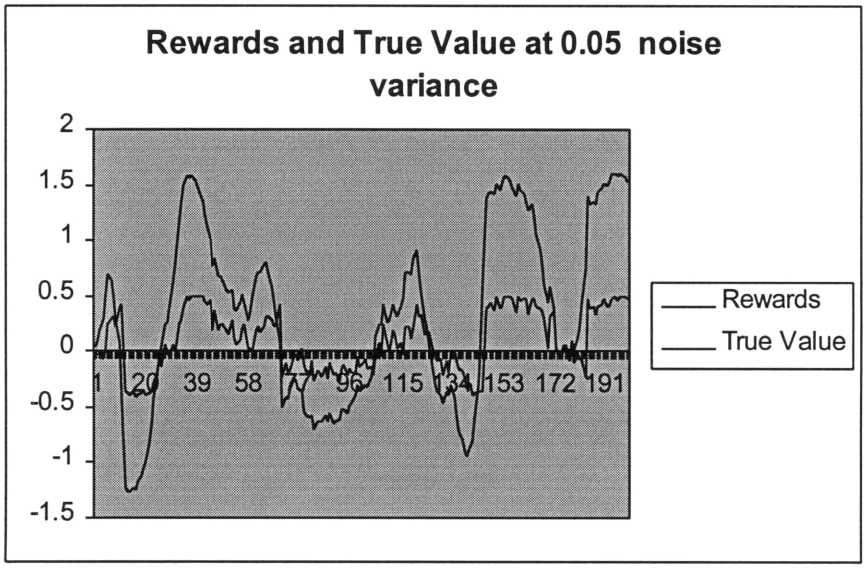


Graph 2

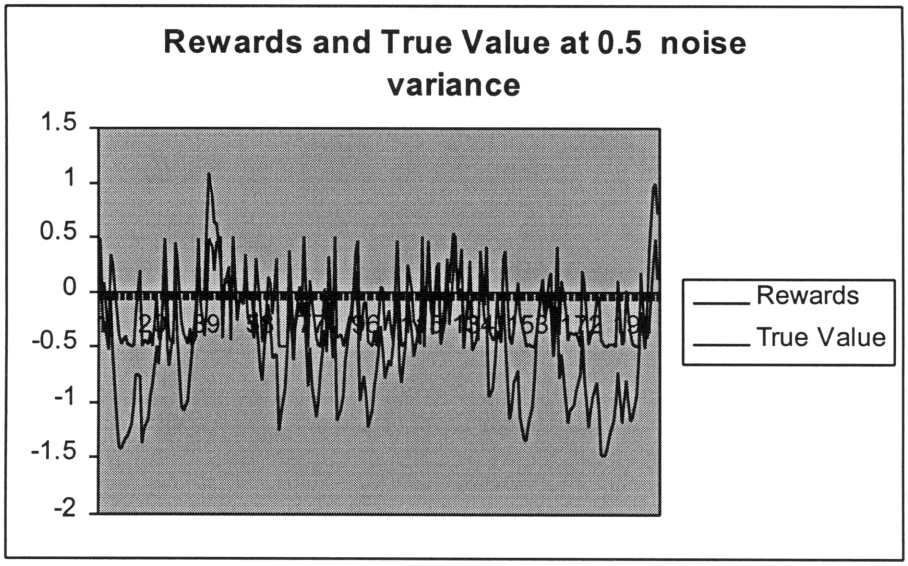
As can be seen in the above graphs, the autoregressive quality of the data diminishes visibly with the increase in the noise level: the data begins to look much

more random. This fact should make policy evaluation a very challenging task in high-noise environments.

The next graph shows policy rewards and estimated *true policy value* for the above data. The true value was estimated by simulating the data equations for 300 data points and computing the actual discounted sum of future rewards for the first 200 points.



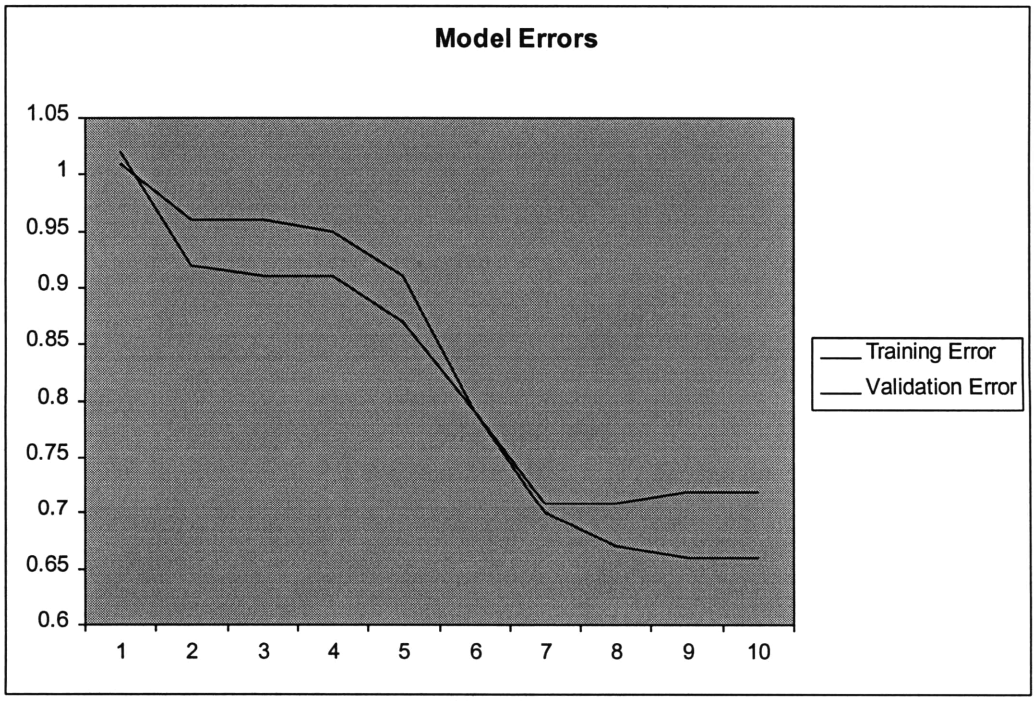
Graph 3



Graph 4

Increase in the noise level also made the rewards look much more random. In addition, it had the effect of shifting down the mean of rewards and consequently of computed values. This is due to the asymmetrical form of the reward function used. In order to account for the above shift, the input and target data was normalized to zero mean and unit variance.

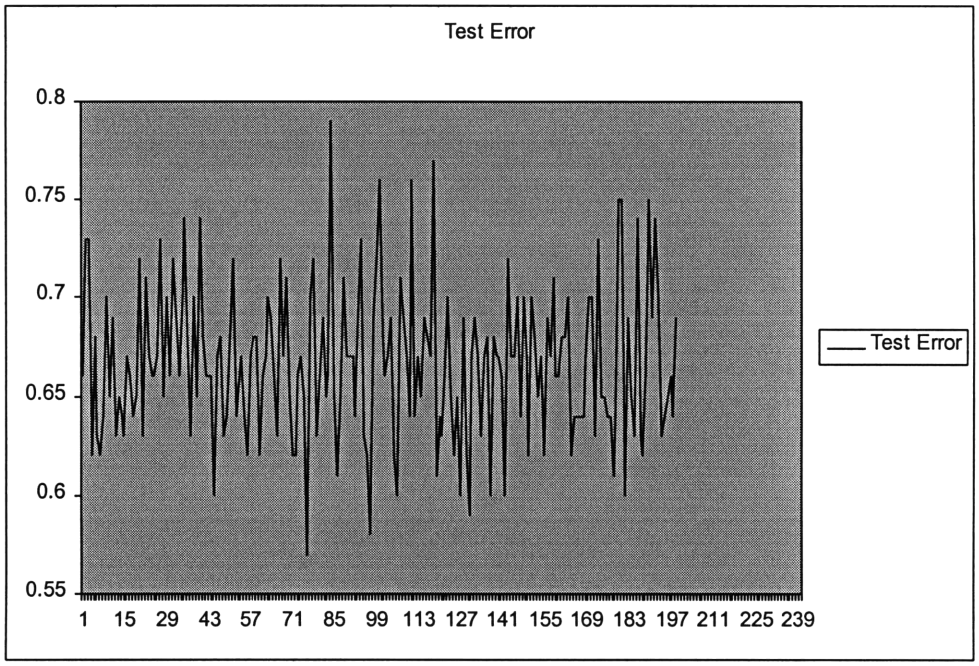
The following results were obtained for a sample training cycle of the model at discounting factor $\gamma = 0.7$. The noise variance in both state and reward equations was 0.05. The graph below shows decrease of the training and the validation error during the training process. The error function used was the logarithm of the likelihood function, as described in chapter 5.



Graph 5

The costs were normalized by dividing them by the cost of the naive value forecast. Due to normalization, the naïve forecast is $V(x(t)) = 0$. Each training epoch took about 15 seconds on an Ultra2 sun workstation. The model error decayed very fast to levels statistically significantly below those of the naive forecast.

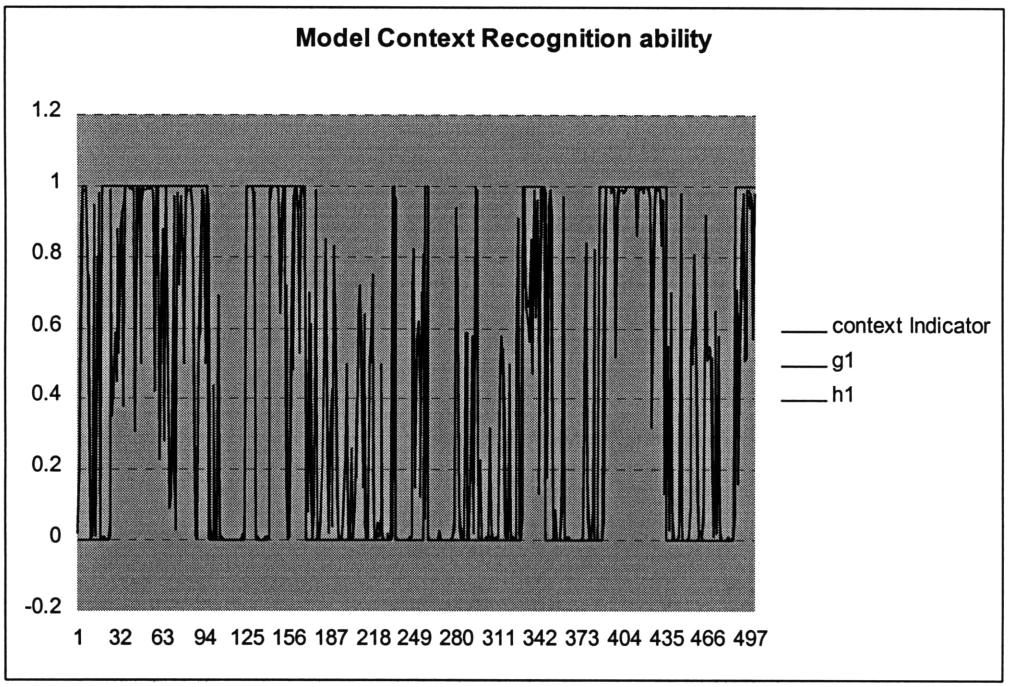
In order to get a statistically valid estimate of the true policy evaluation ability of the trained model, it was applied to different test sets, and its error was observed. The following error graph was obtained.



Graph 6

The 95% confidence level for the mean of the errors in the above graph is 0.67 ± 0.005.

The ability of the model to distinguish favorable contexts in the helicopter's environment from unfavorable is demonstrated in the graph below. For clarity of presentation, a context indicator variable was graphed that is 1 for favorable contexts and -1 for unfavorable contexts. g_1 is the gate's output for expert 1 and h_1 is the weighted error for expert 1 as described in chapter 5.



Graph 7

The above graph shows that the gate has almost converged in its learning, as g_1 matches almost identically h_1. Also, g_1 follows very well the actual testing context, which shows that the gate has learned to recognize contexts. Since h_1 gives an indication of relative expert's errors in each context, the experts had learned to specialize their prediction to contexts.

The procedure of testing the model on many regenerated test sets for estimating the true policy evaluation ability of the model was used for all comparisons described below. The first one compared the proposed gated experts architecture to a similar one used in time series analysis. The field of time series analysis for the most part considers the problem of estimating the immediate future value of a variable based on its past values. The previous usa of the GE architecture [Weigend and Mangeas, 1995] was based on the fact that context changes were apparent in the input data, and therefore the gate was designed to use raw data as inputs. This thesis extends the above problem to the one of forecasting values of the

policy at each time pattern, and the context changes are present only in the way the value is computed. In other words, the context changes are present only in targets rather than both in targets and inputs. In order to deal with this problem, the gated experts architecture in the proposed planning approach used the gate that had expert's past errors as inputs. This was motivated by the idea that since experts' outputs approximate targets, the context information present in targets should also be present in experts' errors.

The following results were obtained after comparing the mean error of two architectures over regenerated test data sets at the 0.05 noise variance.

Mean error for the existing architecture	**Mean error for the proposed architecture**
0.98±0.007	0.67 ± 0.005

Table 1

The performance improvement demonstrated above enables a new usage of the gated experts architecture: estimating long-term policy values rather then just immediate future rewards.

The next test demonstrated the effect on the policy evaluation ability of the model of the discounting factor γ in the reinforcement learning component. Using the intuition from time-series analysis, increase of the discounting factor should make policy evaluation more difficult, since the model in effect will have to forecast policy rewards farther into the future. The following results were obtained.

Discounting factor γ	**95% Confidence level for the mean error at 0.05 noise variance**
0.5	0.73 ± 0.01
0.7	0.67 ± 0.005
0.9	0.82 ± 0.007

Table 2

Results in the above table confirm the ability of the proposed architecture to forecast not only immediate future rewards but also long-term policy values. The larger error at $\gamma = 0.5$ than at $\gamma = 0.7$ contradicted the initial intuition. It demonstrated that for small γ, the smoothing effect of discounting brings more information into the system. However, after some point, the information deteriorates. This can be explained intuitively by the observation that the mean of two independent identically distributed random variables has a smaller variance than that of a single random variable. This is the case when the model is trying to evaluate the policy within the same context. However, when more variables from another context get averaged in, policy evaluation becomes a more difficult task.

In the reinforcement learning component of the proposed planning architecture, the neural network targets were computed by propagating back the expected value of the last data point. The error in the value estimate at time t due to the difference between the true and expected value of the last data point at time T is γ^{T-t}. With $\gamma = 0.7$ and 500 data points, this error falls below 0.01 after last 6% of the data.

The next set of tests compared the performance of the proposed model with more traditional models for policy evaluation. They consisted of combinations of the TD(0) algorithm with different function approximation architectures.

Model Type	**95% Confidence level for the mean error**
Nonlinear Gated Experts	0.67 ± 0.005
Linear Gated Experts	0.74 ± 0.008
Global Model	0.94 ± 0.01

Table 3

Policy evaluation ability of tested architectures with noise variance 0.05.

Model Type	95% Confidence level for the mean error
Nonlinear Gated Experts	0.947 ± 0.002
Linear Gated Experts	0.951 ± 0.003
Global Model	1.026 ± 0.005

Table 4

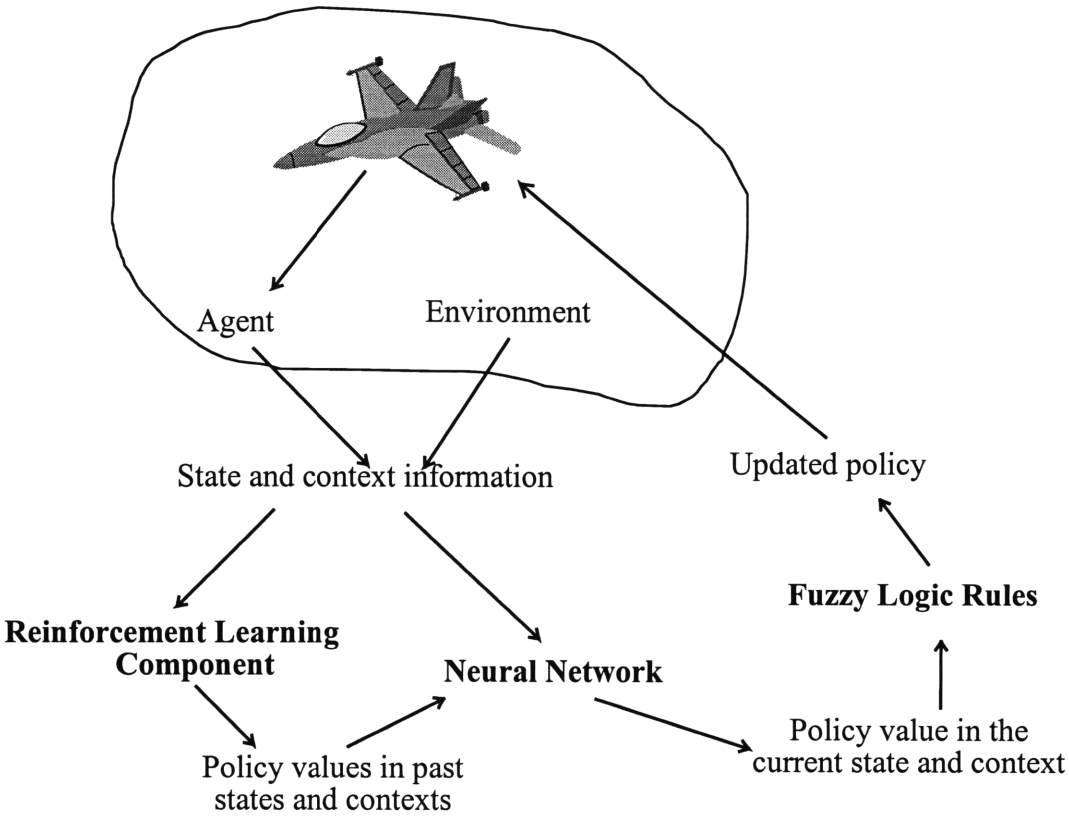
Policy evaluation ability of tested architectures with noise variance 0.5.

The significantly smaller error for the GE architecture over the global model is due to the fact that the experts had a chance to specialize to their contexts and not worry about their errors outside of the context boundary. The outside-of-context errors were decreased by low weights assigned by the gate to expert's forecasts for those time patterns. A global model of each expert, on the other hand, did not have had a chance to exploit the full structure of each context, since it was hindered by trying simultaneously to decrease errors outside of the context. In fact, in high-noise environments, the global model showed significant overfitting, which degraded its performance even below that of the naïve forecast.

The architecture with the linear gate performs better than the global model because it possessed some context recognition abilities. The linear gate also performed only slightly worse than the nonlinear one. This is due to the fact that overfitting is a much bigger problem in higher noise environments. However, after controlling the nonlinear models for overfitting, they can outperform the linear models. In the present case, this control was implemented by the weight decay mechanism, as described in section 6.1. Other more advanced mechanisms can also be used, which is one of the subjects for future research.

Once the superior policy evaluation ability of the proposed architecture was established, the policy updating capability using fuzzy logic rules was tested. In

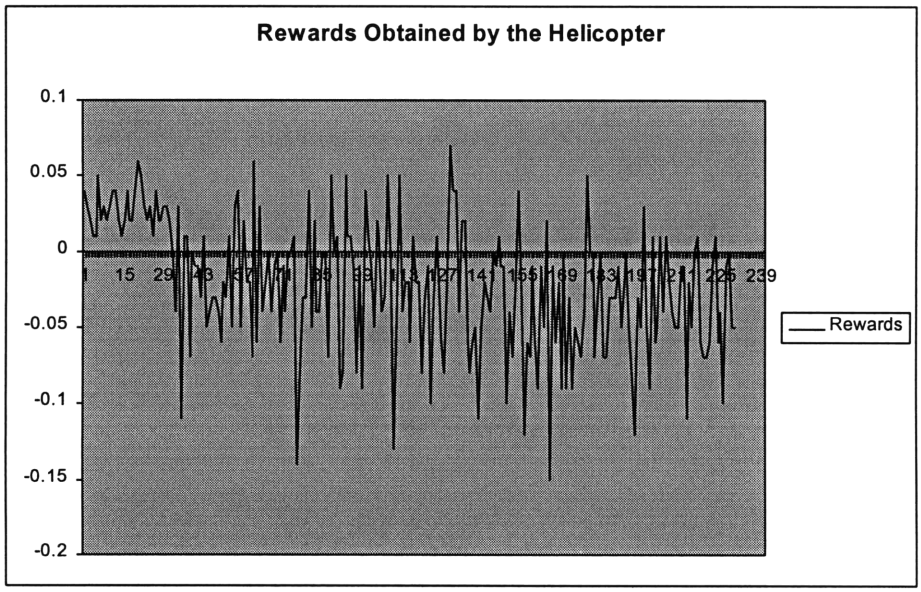
order to save a thousand words, the diagram outlining the components of the proposed architecture is repeated below:



The low-noise environment with noise variance of 0.05 was used for the test described below. The policy updating mechanism was tested in the following manner. First, the GE architecture was trained as usual for policy evaluation on a fixed set of training and validation data. After satisfactory results on the validation set were observed, the training stopped, and the program entered the policy updating mode. In this mode, the values of speed and altitude were generated with varying desired values, which were provided by the fuzzy rules. After the whole test set was generated, the actual sum of rewards was computed as before. This procedure was repeated for a certain number of iterations, during which new test sets were generated. Mean and variance of the sum of rewards were calculated.

In order to compare the above results to the ones with no updating, the program then entered an evaluation mode. The data was generated with desired values for speed and altitude being 0, just as during training. For each test set, the actual sum of rewards was calculated. The mean and variance of this sum were computed after some number of test sets, and was used as the benchmark for comparing with the policy updating results.

The graph below shows the mean of the rewards obtained by the model during training, policy updating, and policy evaluation.



Graph 8

As can be seen in the above graph, helicopter's rewards during the policy updating stage are visibly higher than those during the evaluation stage with no updating. In fact, the 95% confidence level for the mean during the initial stage of updating is 0.03±0.005, and during evaluation with no updating is -0.03±0.006.

Two types of possible errors need to be minimized when choosing appropriate parameters for fuzzy logic policy updating: not changing the policy enough when it is beneficial to do so, and changing the policy too much when it is

not beneficent to do so. These correspond to type I and type II errors in statistical hypothesis testing. These errors should be controlled depending on the risk aversion preferences for the given mission. More specifically, the more extreme a value forecast, the more trust can be given to this forecast. This is based on the idea that forecasts near 0 are more likely to be based on the random noise. Therefore, a higher threshold for policy changing will result in more frequent policy changes with a higher percentage of incorrect decisions. On the other hand, a low threshold will result in less frequent policy changes with a lower percentage of incorrect decisions. However, a very low threshold may result in missing the opportunities to adjust the policy, since the policy will not be updated for many low value forecasts. The number of missed opportunities will depend on the expected frequency of context changes. The length of time window N has a similar effect on the possible errors. A larger N provides more confidence for events observed within this window, since a larger sample of policy values will provide a statistically better estimate for the expected policy value.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

An approach, an architecture and a model with tuned parameters were presented in this thesis for context-sensitive planning by autonomous vehicles. The proposed approach extended the existing planning approaches into the environments of high complexity, uncertainty, and nonstationarity. It consisted of a combination of reinforcement learning, context-sensitive neural networks, and fuzzy logic rules. The reinforcement learning component of the architecture was implemented by the TD(0) reinforcement algorithm. This algorithm was chosen because of its applicability to continuous state spaces with unknown state transition model. The neural network component of the architecture consisted of the gated experts neural network architecture. This architecture was designed to recognize hidden contexts in noisy real data and approximate its targets in a context-sensitive manner. The fuzzy logic component of the architecture consisted of rules for policy updating. The antecedents of fuzzy rules were chosen to correspond to the outputs of the other two architectural components, while the consequent provided information for the path planning policy of the vehicle.

The effectiveness of the proposed approach was demonstrated by applying the corresponding model to the computer-generated data simulating aspects of interest in the environment of autonomous helicopters. The architecture was shown to be superior to existing approaches of combining reinforcement learning with global neural networks. In addition, the proposed approach was compared with a combination of reinforcement learning and neural architectures having linear context-switching models. The results again demonstrated the superiority of the proposed approach, which indicated effectiveness of nonlinear estimation models in complex, uncertain, and nonstationary environments. The above results were consistent both for low and high noise environments, which indicated the

applicability of the proposed planning approach to a broad range of missions of autonomous vehicles.

The following extensions can be made to the proposed planning approach, which would increase its effectiveness and applicability even further.

7.2 Extending the Fuzzy Logic Component

Membership functions for the linguistic variables used as well as the variables themselves in the fuzzy rules can be made data dependent. This will simulate policy updating in a priori unknown environments. The cost function that will be minimized by the fuzzy learning rules will be the negative of this sum of rewards. A standard backpropagation technique can be used for adjusting the mean and variance of the rules.

7.3 Q-learning for Policy Updating

In more general environments where the physical meaning of contexts is unknown, the following procedure can be used to adjust the vehicle's policy. The policy control variables can be made part of the state vector, and a Q-learning type algorithm instead of TD(0) can be used for updating the values of new states. In this case, the values of control variables can be adjusted in every state to give the maximum policy value. This would lead to a nonlinear programming problem in every state, which can be solved quickly for a suboptimal solution. An optimal solution is not required in this case because of the exploration vs. exploitation tradeoff: choosing control variables to maximize the existing valuation scheme would prohibit the vehicle from exploiting other policies that can potentially have even higher values. With the use of Q-learning, when the gate gives more weight to a valuation scheme of a certain expert, actions preferred by that expert will have more weight. This demonstrates the automatic hierarchical planning capability of the proposed framework, in which gate chooses the context that should be paid more attention to and the corresponding expert chooses optimal decisions for that context.

Bibliography

Aamodt, Agnar and Plaza, Enric. (1994) "Case-based reasoning: foundational issues, methodological variantions, and system approaches," Norway, University of Trondheim, Department of Informatics.

Adams, Milton B. (1997) "Draper laboratory experience in real-time planning and autonomous vehicles," C. S. Draper presentation.

Aha, David W. (1997) "The omnipresence of case-based reasoning in science and application," http://www.aic.nrl.navy.mil/~aha.

Asada, M., Uchibe, E., Noda, S., Tawaratsumida, S., and Hosoda, K. (1994) "Coordination of multiple behaviors acquired by a vision-based reinforcement learning," *Proceedings of the 1994 IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 917-924.

Auden, W. H. (1966) *Collected shorter poems: 1927-1957.* p.42, New York: Random House.

Baird, Leemon C., (1995). "Residual algorithms: reinforcement learning with function approximation," http://kirk.usafa.af.mil/~baird.

Barto, Andrew G., Steven J. Bradtke, and Satinder P. Singh, (1993) "Learning to act using real-time dynamic programming," Department of Computer Science, University of Massachusetts.

Baxter, J. W. and Bumby, J. R. (1993) "Fuzzy logic guidance and obstacle avoidance algorithms for autonomous vehicle control," *Proceedings of the 1st International Workshop on Intelligent Autonomous Vehicles,* Southampton, April 1993, pp. 259-264.

Bellingham, James G. (1992) "Capabilities of autonomous underwater vehicles," in *Scientific and Environmental Data Collection with Autonomous Underwater Vehicles,* Report No. MITSG 92-2, MIT Sea Grant.

Bellingham, J.G., C. Goudey, T. R. Consi, and C. Chyssostomidis (1991) "A long range, deep ocean survey vehicle," Submitted to the International Society of Offshore and Polar Engineers.

Bellman, R. (1957) *Applied dynamic programming,* Princeton University Press, Princeton, N.J.

Berenji, H. R. (1996) "Fuzzy Q-learning for generalization of reinforcement learning," *Fifth IEEE International Conference on Fuzzy Systems,* pp. 2208-2214.

Bergmann, Ralph, Hector Munoz-Avila, and Manuela Veloso, (1995) "General-purpose case-based planning: methods and systems," School of Computer Science, Carnegie Mellon University

Bertsekas, Dimitri P. (1995) *Dynamic programming and optimal control,* Athena Scientific: Belmont, Massachusetts.

Bishop, Christopher M. *Neural networks for pattern recognition.* Oxford University Press. 1995.

Blum, Avrim, "On-line algorithms in machine learning," Carnegie Mellon University.

Bonarini, Andrea and Bassso, Filippo. (1994) "Learning to compose fuzzy behaviors for autonomous agents," *International Journal of Approximate Reasoning,* 11:1-158.

Bradtke, S. J. (1993) "Reinforcement learning applied to linear quadratic regulation," *Proceedings of the Fifth Conference on Neural Information Processing Systems,* pp. 295-302.

Chin-Teng Lin and C.S. George Lee, (1996) *Neural fuzzy systems: A neuro-fuzzy synergism to intelligent systems,* Prentice Hall: NJ.

DeBitetto, Paul. (1996) "Red team review of the draper small autonomous aerial vehicle (DSAAV)," C. S. Draper Laboratory internal report.

Doyle, Rory Stephen, (1996) *Neurofuzzy multi-sensor data fusion for helicopter obstacle avoidance,* Ph.D. Dissertation, Department of Electronics and Computer Science, University of South Hampton.

Draper, Denise, Steve Hanks, and Dan Weld. (1993) "Probabilistic planning with information gathering and contingent execution." Technical Report 93-12-04, University of Washington.

Dutta, Soumitra and Bonissone, Piero P. (1993) "Integrating case- and rule-based reasoning," *International Journal of Approximate Reasoning,* 8:163-203.

Gachet, D., Salichs, M. A., Moreno, L., and Pimentel, J. R. (1994) "Learning emergent tasks for an autonomous mobile robot," *Proceedings of the 1994 IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 290-297.

Geyer, E. M., P. M. Creamer, J. A. D'Appolito, and R. G. Rains. (1987) "Characteristic capabilities of navigation systems for unmanned untethered submersibles," Proceedings of the Fifth International Symposium on Unmanned Untethered Submersible Technology.

Harmon, Mance E. and Leemon C. Baird, "Multi-player residual advantage learning with general function approximation,"
http://kirk.usafa.af.mil/~baird.

Haigh, Karen Zita and Manuela M. Veloso, (1997) "High-level planning and low-level execution: Towards a complete robotic agent," *Proceedings of the First International Conference on Autonomous Agents.*

Hayes-Roth, Barbara and Richard Washington, "Practical real-time planning," Knowledge Systems Laboratory, Stanford University.

Honovar, Vasant (1993) "Toward learning systems that integrate different strategies and representations," Department of Computer Science, Iowa State University.

Kaelbling, Leslie Pack, Michael L. Littman, Andrew W. Moore, (1996) "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research* 4, pp. 237-285.

Kaelbling, Leslie Pack, Michael L. Littman, Anthony R. Cassandra, (1997) "Planning and acting in partially observable stochastic domains." Available on the Web.

Koenig, Sven and Reid G. Simmons, "The effect of representation and knowledge on goal-directed exploration with reinforcement learning algorithms," School of Computer Science, Carnegie Mellon University.

Koenig, Sven and Reid G. Simmons, (1997) "Xavier: A robot navigation architecture based on partially observable Markov decision process models," School of Computer Science, Carnegie Mellon University.

Krose, Ben and Joris Van Dam, (1997) "Neural vehicles," *Neural Systems for Robotics,* P. van der Smagt, O. Omidvar (eds.), Academic Press, pp. 271-296.

Kushmerick, Nicholas, Steve Hanks, and Daniel S. Weld. (1995) "An algorithm for probabilistic planning," *Artificial Intelligence,* 76(1-2):239-286.

Lin, L. -J. (1992) *Self-supervised learning by reinforcement learning and artificial neural networks,* Ph.D. Thesis, Carnegie Mellon University, School of Computer Science.

Littman, Michael J., Anthony R. Cassandra, Leslie Pack Kaelbling. (1995) "Learning policies for partially observable environments: scaling up," *Proceedings of the Twelfth International Conference on Machine Learning,* pp. 362-370.

McCallum, Andrew R, (1996) "Efficient exploration in reinforcement learning with hidden state," Department of Computer Science, University of Rochester.

McCallum, Andrew R, (1995) "Instance-based utile distinctions for reinforcement learning," in *The Proceedings of the Twelfth International Machine Learning Conference.* Morgan Kaufmann Publishers, Inc.

McCallum, Andrew R, (1994) "First results with instance-based state identification for reinforcement learning," Department of Computer Science, University of Rochester.

Miksch, Silvia, Werner Horn, Christian Popow, and Franz Paky, (1996) "Context-sensitive and expectation-guided temporal abstraction of high-frequency data," *Proceedings of the Tenth International Workshop for Qualitative Reasoning.*

Mitchell, Tom M. and Sebastian B. Thrun, "Explanation based learning: a comparison of symbolic and neural network approaches," School of Computer Science, Carnegie Mellon University.

Moore, A. W (1992) *Efficient memory-based learning for robot control,* Ph.D. Thesis, Trinity Hall, University of Cambridge, England, 1990.

Murdock, William J., Shippey, Gordon, and Ram, Ashwin. (1996) "Case-based planning to learn," Georgia Institute of Technology, College of Computing.

Park, Jong-Min and Yu Hen Hu. (1996) "Estimation of correctness region using clustering in mixture of experts." *1996 IEEE Conference on Neural Networks.* pp. 1395-1399.

Parr, Ronald, and Russel, Stuart (1995) "Approximating optimal policies for partially observable stochastic domains," University of California at Berkeley, Computer Science Division.

Patek, Stephen D. and Dimitri P. Bertsekas, (1996) "Play selection in football: a case-study in neuro-dynamic programming," Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, LIDS P-2350.

Pawelzik, Klaus, Jens Kohlmorgen, and Klaus-Robert Müller. (1996) "Annealed competition of experts for a segmentation and classification of switching dynamics," Neural Computation 8, pp. 340-356.

Pell, Barney, Erann Gat, Ron Keesing, Nicola Muscettola, Ben Smith. (1996a) "Plan execution for autonomous spacecraft," *1996 AAAI Fall Symposium on Plan Execution.*

Pell, Barney, Douglas E. Bernard, Steve A. Chien, Erann Gat, Nicola Muscettola, P. Pandurang Nayak, Michael D. Wagner, Brian C. Williams. (1996b) "An implemented architecture integrating onboard planning, scheduling, execution, diagnosis, monitoring and control for autonomous spacecraft." NASA working paper.

Peng, Jing, "Efficient memory-based dynamic programming," College of Engineering, University of California.

Plaza, E., Esteva, F., Garcia, P., Godo, L., Lopez de Mantaras, R. (1997) "A logical approach to case-based reasoning using fuzzy similarity relations," http://www.iiia.csic.es.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1992). *Numerical recipies in C: the art of scientific computing.* Cambridge University Press, Cambridge.

Pryor, Louise and Collins, Gregg. (1996) "Planning for contingencies: A decision-based approach," *Journal of Artificial Intelligence Research,* 4:287-339.

Quinlan, J. R., "Combining instance-based and model-based learning," Basser Department of Computer Science, University of Sydney, Australia.

Ricard, Michael J., (1994) "Mission planning for an autonomous undersea vehicle: design and results," C. S. Draper Laboratory.

Schram, G., B. J. A. Krose, R. Babuska, A. J. Krijgsman, "Neurocontrol by reinforcement learning,"

Shanming Shi and Andreas S. Weigend (1997) "Taking time seriously: hidden markov experts applied to financial engineering,"
http://www.stern.nyu.edu/~aweigend

Shannon, C. E. (1948). "A mathematical theory of communication," *The Bell Systems Technical Journal,* 27(3), pp. 379-423 and pp. 623-656.

Simmons, Reid, and Sven Koenig, "Probabilistic navigation in partially observable environments," School of Computer Science, Carnegie Mellon University.

Singh, Satinder P., Tommi Jaakkola, and Michael I. Jordan, "Learning without state-estimation in partially observable Markovian decision processes," Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology.

Smallwood, R. D. and Sondik, E. J. (1973) "The optimal control of partially observable Markov processes over a finite horizon," *Operations Research,* 21:1071-1088.

Sun, Ron (1996) "Hybrid connectionist-symbolic models: a report from the IJCAI'95 Workshop on Connecionist-Symbolic Integration," Department of Computer Science, University of Alabama.

Sutton, Richard S. (1990) "Integrated architectures for learning, planning, and reacting based on approximate dynamic programming," *Proceeings of the Seventh Int. Conf. On Machine Leanring,* pp. 216-224.

Sutton, Richard S. and Andrew G. Barto, "Reinforcement learning: a tutorial," University of Massachusetts.

Sutton, Richard S. and Doina Precup, "Multi-time models for temporally abstract planning," University of Massachusetts.

Sutton, Richard S., Juan Carlos Santamaria, and Ashwin Ram, (1996) "Experiments with reinforcement learning in problems with continuous state and action spaces," University of Massachusetts.

Taylor, Charles and Gholamreza Nakhaeizadeh, "Learning in dynamically changing domains: theory revision and context dependence issues," University of Leeds, United Kingdom.

Teow, Loo-Nin and Ah-Hwee Tan, (1995) "Adaptive integration of multiple experts," *1995 IEEE Conference on Neural Networks.* pp. 1215-1220.

Tesauro, G. J. (1992) "Practical issues in temporal difference learning," *Machine Learning Journal*, Vol 8.

Tsitsiklis, John and Benjamin Van Roy, (1994) "Feature-based methods for large scale dynamic programming," Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, LIDS P-2277.

Tsoukalas, Lefteri H. and Robert E. Uhrig, (1997) *Fuzzy and neural approaches in engineering,* John Wiley & Sons: New York.

Turney, Roy M. (1997) "Determining the context-dependent meaning of fuzzy subsets," *Proceedings of the 1997 International and Interdisciplinary Conference on Modeling and Using Context,* Rio de Janeiro.

Uchibe, Eiji, Minoru Asada, and Koh Hosoda, "Behavior coordination for a mobile robot using modular reinforcement learning," Dept. of Mech. Eng. For Computer-Controlled Machinery, Osaka University, Japan.

Voudouris, Christos, Chernett, Paul, Wang, Chang J., Callaghan, V. L. (1994) "Fuzzy hierarchical control for autonomous vehicles," Department of Computer Science, University of Essex.

Washington, Richard, (1994) *Abstraction planning in real time,* Ph.D. Dissertation, Department of Computer Science, Stanford University.

Waterhouse, Steve and Tony Robinson. (1995) "Constructive algorithms for hierarchical mixtures of experts," *Neural Computation.*

Waterhouse, Steve, David MacKay, Tony Robinson. (1996) "Bayesian methods for mixtures of experts," *Neural Information Processing Systems* 8.

Weigend, Andreas and Morgan Mangeas (1995) "Nonlinear gated experts for time series: discovering regimes and avoiding overfitting," University of Colorado Technical Report CU-CS-764-95.

Widmer, Gerhard and Miroslav Kubat (1996) "Learning in the presence of concept drift and hidden context," *Machine Learning,* 23, pp. 69-101.