Autonomous Helicopter Control using Reinforcement Learning Policy Search Methods

J. Andrew Bagnell, Jeff G. Schneider

Abstract— **Many control problems in the robotics field can be cast as Partially Observed Markovian Decision Problems (POMDPs), an optimal control formalism. Finding optimal solutions to such problems in general, however is known to be intractable. It has often been observed that in practice, simple structured controllers suffice for good sub-optimal control, and recent research in the artificial intelligence community has focused on policy search methods as techniques for finding sub-optimal controllers when such structured controllers do exist. Traditional model-based reinforcement learning algorithms make a certainty equivalence assumption on their learned models and calculate optimal policies for a maximum-likelihood Markovian model. In this work, we consider algorithms that evaluate and synthesize controllers under distributions of Markovian models. Previous work has demonstrated that algorithms that maximize mean reward with respect to *model* uncertainty leads to safer and more robust controllers. We consider briefly other performance criterion that emphasize robustness and exploration in the search for controllers, and note the relation with experiment design and active learning. To validate the power of the approach on a robotic application we demonstrate the presented learning control algorithm by flying an autonomous helicopter. We show that the controller learned is robust and delivers good performance in this real-world domain.**

I. Introduction

RECENTLY there has been a great deal of interest in applying learning for control and planning in robotics and automated systems. "Learning control" entices us with the promise of obviating the need for the tedious development of complex first principle models (see "the curse of modeling" [1]), and suggest a variety of methods for synthesizing controllers based on experience generated from real systems. But reinforcement learning algorithms, particularly applied to real systems where gathering data is costly and potentially dangerous, often include the need to learn some form of system model (either implicitly or explicitly) while determining an optimal policy, so as to be efficient with the data available. Traditional model-based reinforcement learning algorithms make a certainty equivalence assumption on their learned models and calculate optimal policies, using some variant on dynamic programming, for a maximum-likelihood Markovian model. It is our contention that these techniques face serious difficulties in the application to robotics.

First, in nearly all interesting real-world systems we cannot observe the actual state of the system as in a Markov Decision Process (MDP), but rather at best see some noisy function of it, as in a Partially Observed MDP (POMDP). Learning systems should be designed to explicitly account for the resulting violations of the Markov property.

Second, physical systems are often high-dimensional so that it is quite impossible to have data for all parts of state-space. It is also unlikely than any model used by the learning algorithm is capable of capturing all of the subtlety of the real system dynamics, so we would like learning control algorithms to exhibit some degree of robustness to undermodeling. Further, even given a good model, the complexity of building optimal policies typically rises exponentially in the number of dimensions. (The "curse of dimensionality", [2]).

Finally, learning systems, and particularly those operating in the physical world where experiments are costly and time-consuming, must face the well-know exploration/exploitation dilemma. The learning system must trade off: 1) the desire to improve a model by trying out actions and states that have not been well explored (which could improve its overall performance in the future), and 2) the desire to take actions that are known to be good (which yields better near-term performance). The exploration/exploitation problem has received considerable attention. Developing strategies to explore and exploit efficiently is an extremely difficult problem– especially under constraints that are often present in real systems. As an example, consider a helicopter learning its dynamics and a control policy. We want to ensure that it will not crash while learning, or operating under a policy derived from a learned model. Intimately tied to this exploration/exploitation trade-off is the issue of building controllers that are exploration or risk-sensitive.

Recent research in the artificial intelligence community has focused on *policy search methods* as techniques to ameliorate the first two difficulties mentioned above. We argue in this paper that with appropriate performance metrics and algorithms, policy search naturally extends to evaluating and synthesizing controllers under distributions of Markovian models, allowing us to address issues of robustness and exploration.

Drew Bagnell and Jeff Schneider are with Carnegie Mellon's Robotics Institute, E-mail:dbagnell@ieee.org, Jeff.Schneider@ri.cmu.edu.

Although in principle, we can deal with uncertainty in modeling and noisy observations "optimally" by value-iteration of a Partially-Observed Markov Decision Process, the computational complexity is overwhelming. (It is not even known if finding the optimal policy in a discounted POMDP is computable.) The central idea of policy search is to restrict the class of allowed controllers. It has often been observed that in practice, simple structured controllers suffice for good sub-optimal control, and hence recent research has focused on policy search methods as techniques for finding good sub-optimal controllers when such structured controllers do exist. Policy search provides a way to combat the computational complexity issues introduced by the problems above, and requires only a form of simulator, a more compact representation than explicit transitions and costs. By limiting the class of policies to search through, we can potentially much more rapidly find a good policy. Further, limiting the complexity of the controller serves as a form of "regularization". Without structural guarantees, it could take an intractable number of Monte-Carlo roll-outs of a policy on a simulation to evaluate its performance– there could always be some subtlety that is not apparent in any reasonable number of evaluations. This property is captured in theorems relating the uniform convergence of such estimates and the complexity of the searched policy class. [3] Structured policies are very natural in the robotics field as well. It is natural to build restriction we would like on the controller directly into its structure. One can also easily limit the amount of computation required during the control cycle by suitably limiting the complexity of the controller structure. Finally, it is often the case that physical insight leads to good selections of controller class.

A. Previous Work

In [4], safety is addressed by treating learned model uncertainty as another source of noise to be incorporated into the stochastic transitions of an MDP. Good empirical results were obtained, but this method relies on an assumption that model error is uncorrelated through time and space, which is rarely the case.[5] make exploration deliberative and guarantee near-optimal performance in polynomial time. Although this leads to nice theoretical results about the complexity of reinforcement learning, the aggressive exploration such an algorithm encourages is the antithesis of what we would hope for in building safe controllers. The literature on the exploration/exploitation problem in reinforcement learning is extensive. See [6] for further discussion of the problem.

II. Preliminary Setup

We address first the formalism necessary to discuss our results. The measure theoretic details are of little importance and can be ignored with little loss as they are only considered to introduce the notion of re-using samples in evaluations. Consider a controlled stochastic process $X_t(u), 0 \leq t < \infty$ on some probability space $(\Omega, \mathcal{F}, Q)$ taking values in a state space $\mathcal{X}$ and endowed with a bounded reward (equivalently cost) function $-R_{max} < R(x) < R_{max}$ on the state space. Controls u come from a space $\mathcal{U}$ that will typically be taken to be finite, although with suitable restrictions on $X_t(u)$, can also be a taken as more general continuous spaces. For the probability space we will take as a canonical one $[0,1]^\infty$, so as to refer to the bits of the sample space. Note that each next state of the process will be determined by finitely many bits. The stochastic process behaves, after a model M and initial state is chosen by the first bits of ω, as a Markov process with transition kernel $P_{u,M}(x', x)$, or as a Partially Observed Markov Process, which is identical except that controllers only have access to another random variable Y, taking values in an observation space $(O, \mathcal{O})$, that is a measurable $\sigma(X_t)/\mathcal{O}$. We will usually consider controllers mapping X_t to $\mathcal{U}$ (also called strategies or policies here) that come from a restricted class, denoted Π.

For the purposes of this paper, we will consider all off-line simulations to be on a deterministic simulative model [7] where we can sample a typical event, $\omega \in \Omega$ under the distribution Q (the joint distribution of initial states, models, and Markov noise in transitions and observations) and that each such ω can be stored and re-used for the evaluating different controllers. Deterministic simulative models are quite reasonable for model-based computations, but not so for model-free ones. It essentially amounts to being able to reset one's random number generator in a simulation to pick the same event for rolling out different policies in Monte-Carlo policy evaluation, and it provides a critical advantage in optimization, as it ensures that the reward criterion to be optimized will be a function (not noisy). Finally, the assumption of a deterministic simulative model, provides complexity theoretic benefits, in that it allows one to prove uniform convergence of value-estimates to their means in time polynomial in the horizon length.

III. Optimal Policies

A. Performance Criterion

To formalize the notion of building optimal controllers we require a criterion on which to judge the performance of a given controller on a trajectory. A natural one to consider is the (discounted) sum of future rewards achieved under a controller. We denote by $J_\pi(\omega)$ the empirical performance of a policy on a single trajectory:

$$J_{\pi}(\omega) = \sum_{t=0..N} \gamma^t R(X_t(\omega)) \tag{1}$$

$\gamma \in (0, 1]$, $N = (0..\infty)$.

To consider this as a metric on policies, we suggest that policies be ordered by mean trajectory performance, where the expectation is taken with respect to measure Q (including Markov noise and *model distribution*). Note that the initial state, dynamic model, and effects of noise are all specified in the ω. Considering the expectation over model uncertainty and noise is a more complete way to look at model-based reinforcement learning solutions than is usually done when evaluating certainty-equivalence based approaches. We consider the entire posterior distribution on models, and not just the point maximum-likelihood estimate. Finding the optimal controller with this metric corresponds to the Bayesian decision-theoretic optimal controller, when we know the controller *cannot* be changed at a later time due to new information. Formally,

Definition 1: A policy π^* is ϵ near-optimal in Bayesian Stationary Performance if

$$E^*_{\pi}[J^*_{\pi}(\omega)] \geq \sup_{\Pi} E_{\pi}[J_{\pi}(\omega)] - \epsilon \tag{2}$$

B. Connections to robustness and exploration

In many applications it will be important to consider optimization criterion that more explicitly encourage robustness and exploration. We address these issues at length in [8]. Briefly, the central idea for safety and robustness criterion is to consider maximizing the performance on the worst model in a large set of models, or on almost all trajectories the controller executes, so as to, with high-probability, bound the worse-case controller performance. Such robustness procedures when inverted to look at best, instead of worst, performance are similar to heuristic approaches commonly used in experiment design. (For a discussion of the application of stochastic optimization in artificial intelligence and a description of the algorithms mentioned here, see [9].) Algorithms developing controllers to maximize this criterion can be seen as searching for a good experiment to perform to collect information; they are essentially designed according to the "optimism in the face of uncertainty" heuristic. Under this interpretation, the *Bayes optimal stationary controller* described here can be seen as being a version of PMAX– choosing an experiment at the point of largest expected value.

B.1 Convergence of Algorithms

We briefly note that the following theorem on the complexity of evaluating a policy class under the Bayesian Stationary Performance criterion follows immediately from [7]:

Theorem 1: Let a discrete distribution of two-action POMDPs be given, and let Π be a class of strategies with Vapnik-Chervonenkis dimension d = VC(Π). Also let any $\epsilon, \delta > 0$ be fixed, and let $\hat{V}$ be the policy estimates determined by a sampling algorithm using m samples from Ω (the same samples used to evaluate every policy) from scenarios where

$$m = O(poly(d, \frac{R_{max}}{\epsilon}, \log\frac{1}{\delta}, \frac{1}{1-\gamma})), \tag{3}$$

then with probability at least $1 - \delta$, $\hat{V}$ will be uniformly close to V (within ϵ) over all policies in that class. ■

This type of result, while not typically leading to a useful number of samples to actually perform, is encouraging in terms of the tractability of the approach. This result on the polynomial complexity of uniform bounds on the evaluation of performance criterion can be extended to the case of infinite action spaces (with suitable assumption on the complexity of the dynamics). See [7] and [3] for more discussion about policy search and the complexity of uniform bounds on evaluations.

C. Computational Complexity of Achieving Optimality

Proposition 1: Finding the unrestricted stationary memoryless policy that achieves the largest expected reward on distributions over Markovian (or Partially Observed Markovian) Decision Process is NP-hard.

The distribution over models resulting from Bayes estimation in model-based RL leads to a difficult computational problem as we lose the Markov property that makes dynamic programming an efficient solution technique. The problem becomes similar to the one of finding memoryless policies in a POMDP, and thus a reduction similar to [10] proves the result.

D. Sampling Algorithms

Until this point we have deferred the question of sampling from the space Ω. In the case of Bayesian parametric approximators of system dynamics, sampling can be obtained simply by sampling from the posterior of the parameters and then rolling out trajectories as is standard in Monte-Carlo policy evaluation.

However, in many problems in robotics, it has been demonstrated that non-parametric regression techniques admirably serve to model the often highly non-linear and noisy dynamics. [11] These techniques make it impossible to directly sample from the space of possible models. Some non-parametric models like *Locally Weighted Bayesian Regression* do make it possible to sample from a set of posterior local parameters, and hence can generate samples from the 1-step predictive distribution due to model uncertainty. We argue that this, combined with the ability to re-estimate the model in the Bayes-optimal way, is sufficient to create arbitrary length trajectories that are independent samples from the n-step predictive distribution. If a regression algorithm like LWBR is **not** a Bayes optimal estimator, the technique described in this section provides biased n-step samples that we hope are close approximations to the ideal samples.

Algorithm 1 (N-step predictive sampler) Algorithm to generate samples from the N-step predictive distribution of a learner with 1-step predictive distributions

1. Generate a sample state transition from the 1-step predictive distribution and update the current state
2. Update the learned model using the generated state transition as if it were a training point observed from the real system
3. Repeat to 1 until a termination state is entered or effective horizon is reached (For the analysis below assume we repeat n times.
4. Reset the learned model back to the original model

■

If our estimator were optimal in the Bayesian sense, we would expect that iteratively re-estimating the model using generated samples from the model, as the algorithm above suggests, would indeed allow us to sample from the n-step predictive distribution.

Theorem 2 (Sufficiency of 1-step predictive learners) If model M in algorithm (1) can be recursively updated in the Bayes-optimal way, the trajectories generated by the algorithm (1) are independent samples from the n-step predictive distribution.

Proof: We argue by induction. Consider the two step predictive distribution:

$$p(X_2, X_1|X_0, \mathcal{T}) = p(X_2|X_1, X_0, \mathcal{T})p(X_1|X_0, \mathcal{T}) \quad (4)$$

where $\mathcal{T}$ is the observed data used to build the model. For a discrete model set,

$$\begin{aligned} p(X_2|X_1, X_0, \mathcal{T}) &= \sum_{\mathcal{M}} p(X_2|X_1, X_0, \mathcal{T}, M')p(M'|X_1, X_0, \mathcal{T}) \\ &= \sum_{\mathcal{M}} p(X_2|X_1, M')p(M'|\mathcal{T}') \end{aligned} \quad (5)$$

where $\mathcal{M}$ denotes the discrete class of models to be estimated from the data. The second distribution in each summation is just the posterior model M'; that is, the distribution over Markov models conditioned on the observed data and the transition from X_0 to X_1. But then the final equation shows that $p(X_2, X_1|X_0, \mathcal{T})$ is just another one-step distribution from the new distribution of models $P(M')$, simply the learned model under the old data and the new observed transition.

Fig. 1. The CMU Yamaha R50 helicopter in autonomous flight.

Similar results can be shown with more technical detail in the case of other model distributions. It follows then from the law of composition that if X_1 is first drawn i.i.d $p(X_1|X_0, \mathcal{T})$ and then X_2 is drawn from $p(X_2|X_1, X_0, \mathcal{T})$, the pair is i.i.d from the joint predictive distribution.

■

IV. Experimental Results

There is ample room to apply the techniques developed in the machine learning community to the problems in the control of autonomous helicopters. Autonomous helicopter control is difficult as the dynamics are unstable, non-minimum phase, have large delays, and vary a great deal across the flight envelope. In this section we detail some of the results from applying the policy search methods described in the previous sections to the problem of the flight control of an autonomous helicopter.

A. Dynamics

We begin by specifying the problem. To provide a manageable first goal in applying policy-search to the helicopter, we considered only the so-called "core dynamics" of the helicopter, the pitch, roll, and horizontal translations. The dynamic instabilities are known to lie in these dynamics, and control of these is therefore paramount. [12] Existing proportional-derivative (PD) controllers, tediously tuned by the helicopter team, were used on the yaw-heave dynamics. From a high-level, the goal will be the regulation of the helicopter hovering about a point, or a slowly varying trajectory. This will be formalized as a cost function to be optimized.

B. Modeling

Modeling a dynamical system is always challenging. To learn the dynamics of the helicopter, we chose to implement a LWBR state-space model of the following form (a locally affine model):

$$z\overrightarrow{x} = \mathbf{A}(\overrightarrow{x})\overrightarrow{x} + \overrightarrow{ref}(\overrightarrow{x}) + \mathbf{B}(\overrightarrow{x}) \begin{bmatrix} \delta_{lon} \\ \delta_{lat} \end{bmatrix} \quad (6)$$

$$\overrightarrow{x} = [x, v_x, \theta, \theta - \frac{1}{z}\theta, y, v_y, \phi, \phi - \frac{1}{z}\phi]^T$$

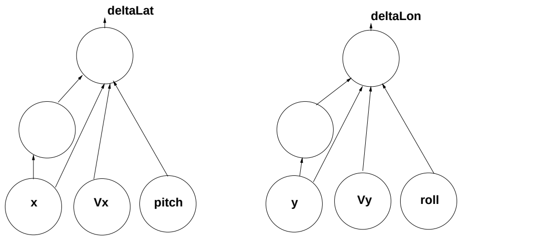

Fig. 2. A graphical depiction as a neural network of the structure of the policy used to control the helicopter.

where z is the forward-shift operator. The inputs, δ_{lon} and δ_{lat} reflect the cyclic controls of the helicopter. The state variables x and y refer translational deviation of the helicopter from its set point. Massive cross-validation was applied to determine appropriate kernel widths. Data was collected from pilot tele-operation of the helicopter. This data was recorded off the Kalman state-estimator at 100Hz and down-sampled to 10Hz. The down-sampling introduces aliasing into the data due to the higher-order dynamics of the helicopter, but has the advantage that it reduces the apparent delay in controls applied caused by the unobservable rotor and actuator dynamics. We hope to still capture much of the principle behavior of the helicopter with the lower frequency model. Interesting future work would involve building a different state-space model capable of capturing the higher frequency dynamics the helicopter demonstrates.

C. *Controller design*

C.1 Controller structure

In proposing an initial controller structure, we looked towards simple controllers known to be capable of flying the helicopter. To this end, we proposed a simple, neural-network style structure (see Figure (2)) that is decoupled in the pitch and roll axis, and about equilibrium is similiar to a linear PD controller. There were 10 parameters to modify in the controller structure, corresponding to the weights between output nodes and parameters in the sigmoidal functions at the hidden and output layers. This is a fairly simple controller that leads to a policy differentiable in its parameters and nearly linear about equilibrium. Because of the hidden layer unit, it is able to adapt to large set-point shifts in the position variables, unlike a linear one.

C.2 Optimization

For the purposes of optimization, we maximized the Bayesian Stationary Performance criterion. It has previously been demonstrated that this criterion (or rather the approximation of it given in [4]) typically leads to controllers that are neither too conservative, nor as aggressive as that obtained by a optimizing a maximum likelihood model. A variety of cost criterion were implemented, each leading to different (although mildly so) controllers. A typical example was the quadratic form:

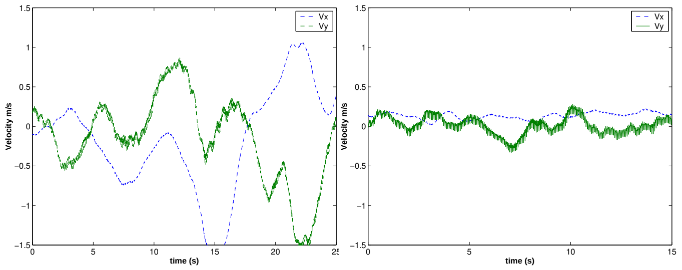

Fig. 3. Data logs from the R-50 demonstrating performance hovering under (left) a highly trained pilot's control, and (right) the neural-net controller built by the robust policy search methods.

$$x^2 + y^2 + \dot{x}^2 + \dot{y}^2 + .0001 * \delta_{lat}^2 + .0001 * \delta_{lon}^2 \qquad (7)$$

Quadratic forms in the position variables are typically not reflective of our real performance goals. A better reflection of our preferences in control is one that emphasis stability in velocities and angles rather than placing huge penalties on modest deviation from nominal set point– especially as we will expect the controller to perform well even when started quite far (as distant as 25 meters) from the set point. To reflect this preference, our immediate cost criterion on the position variables is computed linear in the magnitude of the state variable, or of a form like the following:

$$10\frac{x^2}{x^2+1} + 10\frac{y^2}{y^2+1} \qquad (8)$$

Finally, we assigned a large penalty for large (10^6) for state-variables that were outside the space of the data we had observed.

After establishing the cost criterion, we considered the task of optimizing the parameters. Trajectories were rolled out using the sampling technique described in algorithm (1) for the LWBR model. Typical policy evaluations were 30 trajectories of horizon length 500 with discount factor $\gamma = .995$. The amoeba (simplex) optimization technique [9] was used to modify the controller parameters and guide the search for optima. Random restarts were applied to initial weights to allow the optimizer to find a reasonable solution. Note that is possible to use differentiable optimization techniques instead, if we were willing to smooth the "out-of-bounds" penalties introduced on the state-variables.

D. *Validation*

Initial validation experiments were performed on a linear model of the rotorcraft about hover given in [12]. Good performance on this model was encouraging as it is significantly higher-dimensional (14^{th} order) and larger bandwidth model than that obtained using the locally weighted regression technique described here, and was developed by a different set of techniques. In particular, to formulate a state-space model [12] uses the U.S. Army developed CIFER system, designed specifically for full-scale rotorcraft identification. CIFER performs system identification in the frequency domain using the Chirp-Z transform, and treats the state space identification as an optimal matching problem. This approach has been validated numerous times, and takes advantage of the known physics of the helicopter. Further, because of the state space structure, CIFER explicitly models rotor dynamics (which introduce a significant delay into the system dynamics). This modeling enables Mettler et. al. to capture higher frequency dynamics than we can hope to. However, the frequency domain approach is fundamentally a linear one, and thus one can only capture dynamics in a small part of the flight envelope. Further, the inflexibility of the state-space model in CIFER forces the response into a particular structure that one cannot be sure is accurate for all rotorcraft, particularly ones of dramatically different scales then that for which CIFER was originally designed. Despite these reservations, it is apparent that [12] get excellent results for the hover centered model.

It is interesting to note that controller developed by policy search on the maximum likelihood model had highly oscillatory (nearly unstable) performance on the linear simulator. The controller learned on the distribution of models, in contrast had significantly lower loop gain.

After a degree of confidence was achieved by simulation on the model, the controller was ported to Carnegie-Mellon's autonomous R-50 helicopter control computer. The estimation-control loop on-board operates at 100Hz (as opposed to the simulation 10Hz). To ensure the controller wasn't operating outside the modeled bandwidth, a first order low-pass digital filter was implemented on the control outputs.

The helicopter was then test flown. The results were encouraging, and demonstrate that the simple policy search technique can generate controllers that are applicable to robotic systems. Despite quite windy conditions the rotorcraft was able to track moving set points and reject strong gusts. Figure (3) shows typical performance during the flight, contrasting the hovering of a highly trained human pilot with the controller obtained using the safe learning control methods described above.

V. Conclusions

Our current work establishes a framework for the development of structured controllers sensitive model uncertainty and then demonstrates the viability of the approach on a difficult physical control problem. Future research directions include more sophisticated control of the rotorcraft to exercise more of the flight envelope. Another interesting area to pursue is the online implementation of the inner-loop policy search to dynamically reconfigure the controller. This is particular interesting in the context of error-recovery. In the event of a failure of some sub-system of the helicopter, it will be critical to, based only on very limited experience after the fault, quickly search for a viable controller. We will also extend this work to apply more efficient techniques in the inner loop search, including faster optimization and using reward shaping in a principled way. Finally, we will continue the investigation into explicit criterion for exploration and risk sensitivity using policy search described in [8].

Acknowledgments

The authors gratefully acknowledge enlightening conversations with Bernard Mettler and Andrew Ng, and particularly the help of Omead Amidi in the implementation and experiments with the helicopter. Finally, the authors thank Chuck Thorpe for his support throughout the research. Drew Bagnell was supported by Robotics Institute and National Science Foundation Fellowships.

References

[1] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming.* Athena Scientific, 1996.

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

[3] M. Kearns, Y. Mansour, and A. Ng, "Approximate planning in large pomdps via reusable trajectories," in *Neural Information Processing Systems 12*, 1999.

[4] J. Schneider, "Exploiting model uncertainty estimates for safe dynamic control learning," in *Neural Information Processing Systems 9*, 1996.

[5] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," in *International Conference on Machine Learing*, 1998.

[6] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[7] A. Ng, D. Harada, and S. Russell, "Pegasus: A policy search method for large mdps and pomdps," in *Uncertainty in Artificial Intelligence, Proceedings of the Sixteenth Conference*, 1999.

[8] J. Bagnell, J. Schneider, and A. Ng, "Robustness and exploration in policy-search based reinforcement learning," tech. rep., Robotics Institute, Carnegie Mellon University, 2000.

[9] A. Moore and J. Schneider, "Memory based stochastic optimization," in *Advances in Neural Information Processing Systems (NIPS-8)*, 1995.

[10] M. Littman, *Algorithms for Sequential Decision Making.* PhD thesis, Brown University, 1996.

[11] C. Atkeson, "Using locally weighted regression for robot learning," in *Proceedings of the 91 IEEE Int. Conference on Robotics and Automation*, April 1991.

[12] B. Mettler, M. Tischler, and T. Kanade, "System identification of small-size unmanned helicopter dynamics," in *Presented at the American Helicopter Society's 55^{th} Forum*, 1999.