Path Integration for Stochastic Control and Machine Learning

Drew Bagnell

May 4, 2002

Abstract

Planning and control under uncertainty is a central problem in robotics and artificial intelligence. Typically phrased in terms of Bellman equations and value-functions, the stochastic control problem also admits a "first-principles" definition as one of optimizing a controller's expected reinforcement over possible trajectories of a system. In Quantum Field Theory, a similar situation arises termed "path integration" where quantities of interest are computed as expectations (under a complex measure) by summing over all possible trajectories (or "paths") a system may take. Although the understanding of the control problem as one of maximizing an expectation is hardly novel, I contend that this rephrasing in terms of path integration may be profitable. Much recent research in the engineering, statistics and artificial intelligence communities can be viewed as sophisticated techniques for performing integration. I propose that these techniques may be very useful in control. As preliminary results, I present novel algorithms based on importance sampling and Minka's [30] Expectation Propagation.

Defining the control problem in terms of distributions over paths also broadens the class of problems that can be considered. Robotic applications motivate the development of techniques that acknowledge the inevitability of under-modeling and model uncertainty. This proposal presents two novel models that seek directly to minimize the risk and impact of brittle controllers. The first extends the notion of optimization of the path-integral to an adversarial game where nature chooses a possible system (including highly non-Markovian ones), and hence a distribution over trajectories, from some large set that models our uncertainty. An efficient dynamic-programming type algorithm is provided to solve this class of problems.

The second model captures uncertainty in learning a dynamic model. Large state-spaces invariably lead to sparse training data and the certainty-equivalent approximation of ignoring the resulting uncertainty may lead to controllers over-tuned to a particular, inaccurate model. A more Bayesian approach to finding good controllers is suggested by integrating out the uncertainty in the system model. This technique was used to learn robust controllers for Carnegie Mellon's autonomous helicopter using limited training data. The resulting controller is, to the best of my knowledge, the first successful application of reinforcement learning to the difficult problem of rotorcraft control.

Introduction

The theory of quantum mechanics may be discussed in an elegant formulation due to Feynman [13] known as *path-integral* or *functional* quantization. In this approach, quantities of interest to the physicist, such as the *propagator* or *density matrices*, are computed by summing over all possible paths a particle may take weighed by a (complex) measure on each of those paths. This approach is often regarded as pedagogically superior to the *canonical* (Hilbert-space) quantization program as it makes manifest symmetries critical to the physical system and the stochastic nature of quantum mechanics. In many cases it is also the only known successful approach to quantization [40] and further, establishes deep connections between Quantum Field Theory and Statistical Mechanics.

A foundational problem for artificial intelligence is that of controlling systems under uncertainty. This work proposes *path-integration* (albeit using standard probability measures) as a foundation for this problem of stochastic control; we view the problem of control or planning under uncertainty as one of optimizing the value of a controlled path-integral over a set of policies. Specifically, we will define the value of a controller as the expectation over paths (with distribution induced by the controller) of the cost (reward) along a path. The more standard formulation in terms of *value-functions* as fixed points of *Bellman equations* will arise for Markov Decision Problems (MDPs) from the one proposed here in a way analogous to the way Hamiltonian dynamics arises from the Lagrangian formulation of mechanics.

Understanding the problem of stochastic control as related to integration is not novel, as many researchers have defined the control problem as maximizing an expected reward over time. Rather, it is my hope that the consistent and explicit interpretation of the problem in this light will pay dividends in new computational and analytic tools. Further, the definition of the stochastic control problem in terms of path integration broadens the class of control problems beyond the standard MDP (or even Partial Observed MDP) formulations. Any system that, given a controller, defines a unique probability measure on paths of a system is admissable, including problems involving uncertain models and involving continuous time and state.

This works suggests that although many powerful algorithms (e.g. Value Iteration) fuse the notions of computing the value of a controller with its optimization (and there *are* deep synergies here that are not to be ignored), it is profitable to tease apart these processes. A great deal of recent work in the communities of engineering and information sciences (e.g. [50] [30] [35] [21]) can be viewed as sophisticated analysis and algorithms of the kind of high-dimensional integrals that path-integration will require. I will attempt to adapt and extend this work to control.

In this research proposal, I will first introduce the standard stochastic control models used in robotics, operations research, and AI and discuss how well known algorithms may be interpreted in terms of path-integration. Next I will demonstrate two new computational techniques for evaluating controller performance. Finally, I will discuss two novel models of control under uncertainty that generalize MDP models to handle model uncertainty and non-Markovian dynamics.

2. Path Integration

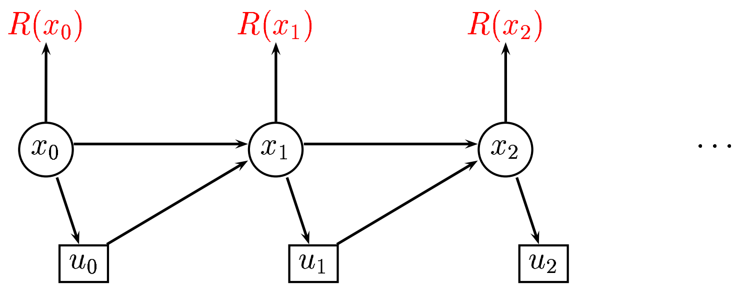
The control problem is fairly intuitive but some formal definitions and notation will aid the discussion. A stochastic control problem consists of a space of paths (also called system trajectories) Ξ, a random variable ξ taking its values in the space of paths with distribution over paths $P(\xi)$ that is a function of a sequence of controls and $< u_t >_{t\in\{0,..,T\}}$ from a space $\mathcal{U}$. We also typically have a sequence of outputs (observations) $< y_t >_{t\in\{0,..,T\}}$. measurable with respect to ξ. A controller is a feedback-type strategy that maps the history of observations $< y_t >_{0,..,\tau}$ to a distribution over controls u_τ, or more formally is a random variable measurable with respect to the observation history. We will at times refer to state-variables x_i^t of the system that compose a path ξ. Unless stated otherwise these may not be "states" in the physics or control theoretic sense of rendering the past and future independent, but rather are convenient component variables of ξ. The goal of the control problem is to find a controller that maximizes a reinforcement function $R(\xi)$ defined over the path space. In particular, we will have controllers parameterized by θ, which induce a distribution over paths $P_\theta(\xi)$ and we seek to maximize the following expectation, termed the path integral (of the reinforcement function $R(\xi)$):

$$J_\theta = \sum_{\Xi} P_\theta(\xi) R(\xi) \tag{1}$$

This definition of course is given in terms of a discrete set of paths. It becomes a true integral in the case of a continuum of paths. In the limit of arbitrarily fine partitions of time, we recover the true functional integration of Feynman [13] (at least for statistical physics systems where we have a standard probability measure). There are true subtleties in this case, and the integral is usually defined as the limit on a fine, discrete lattice or with some momentum (Fourier transform) cutoff to regulate what is otherwise a divergent process. Nevertheless, the discrete case is a powerful intuitive device even for the continuous case. We now discuss some of the better known models that are special cases of the path-integral definition.

2.1. Markov Decision Processes

The Markov Decision Process (MDP) forms perhaps the most common model in control, operations research, and machine learning. Although we discuss it in the discrete state case, it extends naturally to continuous state (like the linear state-space models that dominate control theory). We have a set of states from a space denoted $\mathcal{X}$ and controls denoted $\mathcal{U}$. A reinforcement function $R(x)$ is defined over the state space. (Trivial extensions exist to the case when R is defined on the a state, a control, and the next state; although for most of this document it is easier to discuss it in the simplified form.) States are formally so in the sense that they obey the Markov property: x_{t+1} is conditionally independent of the history of the process given only the control and x_t, the previous state. In the language of graphical models [22] the MDP is a directed chain of states decorated with control and reinforcement nodes:



The term Markov Decision Problem refers to the process as well as a performance criterion: the most common one is simply additive in time:

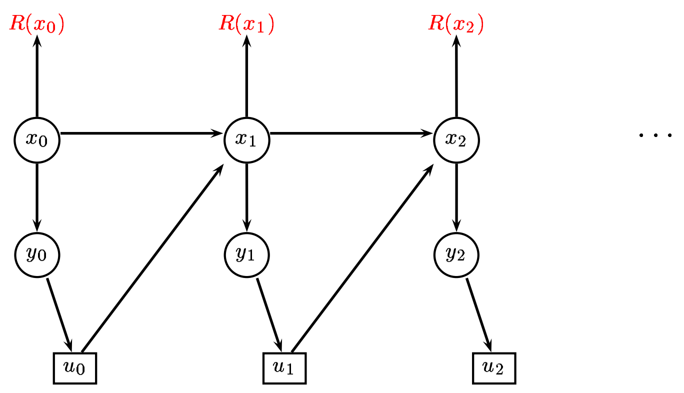
$$R_{path}(\xi) = \sum_t R(x_t)$$

The transition model describing $P(x_{t+1}|x_t, u_t)$, reward function, together with a distribution over states at $t = 0$ and a controller uniquely fixes the value of the path integral. The well-known discounted criteria (linear in time with exponentially decaying weight) can be easily captured under the above definition by simply adding a transition to a zero-reward absorbing state with probability $1-\gamma$, where γ is the discount factor. An excellent elementary treatment of MDPs and the dynamic programming algorithms most commonly used in their solution can be found in [42]. Convergence of the path-integral is guaranteed for systems that almost-surely reach a terminating state (or terminate in finite-time), which includes, of course, the discounted case.

The standard approach to the solution of MDPs is to leverage results known as Bellman's [8] equations, which establish a relationship between path integrals starting from different initial states. The Bellman equations are either directly iterated as fixed point equations, or used in a linear program to solve directly for the optimal controller.

2.2. Partially Observed MDPs and other extensions

Although simple and understandable, the MDP is too simple a model for most of the processes we are interested in. The most realistic and natural extension of the MDP is the Partially Observed MDP [25], in which the underlying state-structure is preserved while only some random variable (denoted here y_t), measurable with respect to the state at each time step, is observed. Below is depicted the graphical model of a POMDP with a memoryless controller:



There are elegant structural properties to the optimal controller under the criterion of maximizing the total (sum) reinforcement. Most importantly, POMDPs (*only* under this particular criteria!) reduce to *belief-state MDPs*. That is, the Bayes-optimal filter's distribution over possible underlying states at each possible time step constitutes a sufficient statistic for control, and optimal controllers become simply memoryless policies mapping the output of this filter directly to the space of controls. This elegant result lies at the heart of the Kalman-filter/Linear Quadratic Regulator optimality (combined with the quadratic cost function). Despite this appeal, with the exception of special cases and tiny, toy discrete problems, exact POMDP solution techniques have thus far been computationally useless. Moreover, it is understood now that the problem is fundamentally a difficult one; in fact, in the average reward case (eminently tractable in the MDP case [42]) finding the optimal control is simply non-computable.

A review of solution techniques can be found in [34]. Many recent techniques, falling under the rubric of "policy-search" claim to be POMDP solution algorithms. While this is true, it is to some extent misleading. Many of these algorithms optimize controllers from some limited class applied to essentially any controlled stochastic process, and so the qualifier POMDP may be overly restrictive.

2.3. Analytic models in Continuous-Time Finance

The standard approach to continuous-time finance is the development and solution of stochastic differential equations [46]. [1] Recently, members of the physics community have proposed an approach completely analogous to the one described here to financial calculations, replacing the Ito calculus with one based on integration [20]. Besides being much more transparent this approach opens the door to research in applying the techniques developed in the information engineering and

[1]The theory of continuous time stochastic processes has a well-deserved reputation for being difficult and opaque to the uninitiated. This is largely due to its measure-theoretic foundation which introduces a great deal of complication. The limit of discrete stochastic processes is treated as only an intuition rather than a rigorous definition. The path-integral approach makes it relatively easy to side-step these complications by directly defining a measure on a space of paths.

sciences community to this field, one of immediate and global significance. In the thesis, I will further explore these connections.

3. Novel Computational Tools

The vast majority of problems of interest in control and AI are not exactly solvable (either analytically or efficiently computationally) as some of the above are, usually because they involve very difficult path integrals and non-convex optimization problems. Rather, we must consider a variety of approximations. There are two well-studied approximations used in the AI literature. First is the use of approximate dynamic programming (by way of approximate value-functions) and the other is the use of Monte-Carlo "roll-outs" (trajectory simulation). These are often combined to form a stochastic iterative approximate dynamic programming algorithms. This set of techniques form the basis of "Neuro-dynamic programming" [10], which has scored at least one spectacular success in learning a world-class backgammon playing algorithm. [48]

Both methods, unfortunately, have their difficulties. Perhaps the central fallacy of the value function approach is that an improving value-function implies an improving policy [7]. On the contrary: in a wide variety of practical applications it has been observed that approximate value functions, while steadily reducing their approximation error (i.e. Bellman residue [4]), often initially improve a policy, but later the policy grows much worse. In other examples, value function based methods diverge– giving arbitrarily poor approximations. [17] Only a limited number of methods have sound convergence properties and fewer still any convexity or uniqueness results. [10] [17] [4] Finally, and perhaps most importantly, value function methods are fundamentally limited to MDPs. These methods have great promise nevertheless as heuristic "critics" to drive the search for a good policy. This is a very open and potentially fruitful area of research.

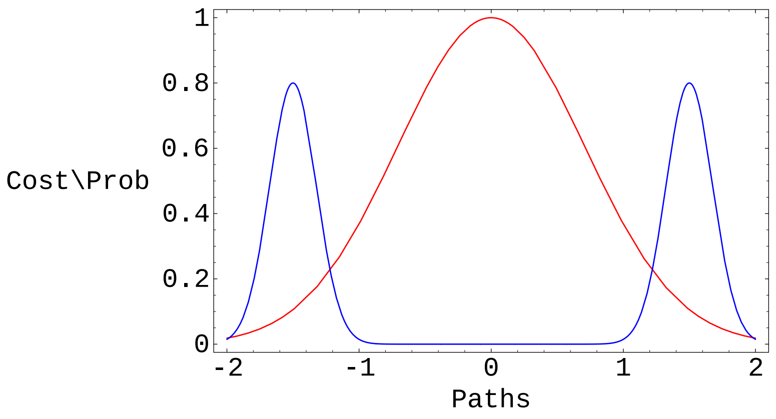
Monte-Carlo (MC) "roll-out" methods for policy evaluation have gained a great deal of popularity recently as it became fully understood the generality of the problems they could solve. Although a great deal of research has indicated the power of this approach, standard simulation based optimization of a controller is perhaps overly general ("Monte Carlo may be a sledgehammer so heavy you couldn't hope to pick it up." [44]). Principally, the generic simulation technique suffers from high variance and noisy evaluations. This makes optimization a particularly difficult problem, especially in regions of controller space where the gradient is small. Potentially it may take a tremendous number of samples to get a good estimate of J_θ or its derivatives. The next section investigates some novel approaches to solving this problem for control.

3.1. Monte Carlo

The first class of techniques to be considered are those that form randomized "any-time" approximations to the path-integral J_θ. It is natural to wonder whether there are any benefit to be gained over naive sampling methodologies; the standard simulation or "roll-out" technique generates exact samples from the distribution over paths and we can easily compute the path-integral as:

$$\frac{1}{N}\sum_{i=1..N} R(\xi_i)$$

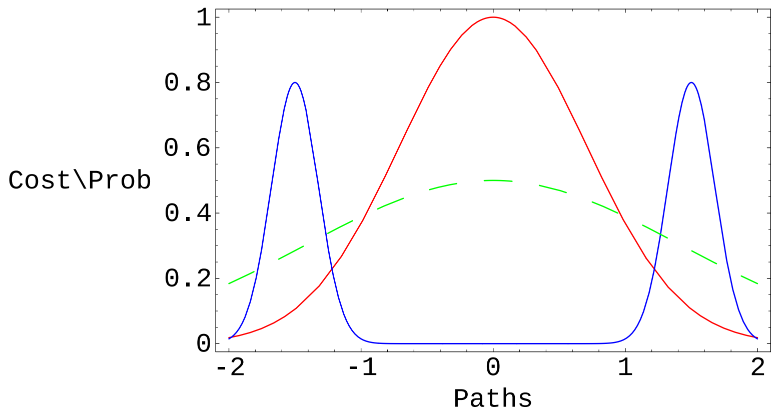
where $R(\xi_i)$ indicates the reinforcement (cost or reward) incurred along each sample path. The key insight is that we have "perfect" samples from the "wrong" distribution. The following diagram abstractly captures the problem that is overwhelming common in these problems:



The single Gaussian bump indicates the distribution over paths, while the double bump indicates the cost function along paths. The poor match indicates that when trajectories are sampled they will rarely capture the important areas of the cost function. This becomes very problematic in high dimensions. Although this example is a one-dimensional real integral, it still captures the notion that the cost along paths may be very poorly matched to the sampling distribution of paths. As a result, it may take a tremendous number of samples to achieve a good approximation. We seek then, a Monte-Carlo approach that lowers the *variance* of our estimate. One approach very recently tried in the literature is that of importance sampling. In importance sampling, one samples from an approximate distribution and corrects the sample average by a factor

$$\frac{p(\xi)}{q(\xi)}$$

(where $p(\xi)$ is the true sampling distribution and $q(\xi)$ the substitute). Below, as a dashed line (in green) we see a substitute sampling distribution that captures more of the tail behavior of the cost function.



Although trivial in the toy example depicted, it may be quite difficult on the high dimensional path-integrals of interest to find and sample from a useful distribution. In general, to minimize the variance of the estimate of the path-integral there is an optimal distribution to sample from:

$$q(\xi) = \frac{|R(\xi)|p(\xi)}{Z}$$

Unsurprisingly, this distribution is quite difficult to sample from: the normalizing constant (the partition function Z) *is* the quantity we wish to estimate (at least for $R() \geq 0$)! So it seems we have converted a problem of difficult integration to one of difficult sampling. In general, this may gain us nothing; however, the conversion motivates a slew of new approaches based on using problem-specific samplers instead of the completely generic roll-out type.

3.1.1 Importance Sampling with a Shaping Function Another well known approach to lowering the variance of roll-out estimates is that of "shaping" the reward function– that is, changing the reward function to lead to lower variance estimates of the value. An elementary proof (without appeal to Bellman's equation) based on the path-integral formalism proves an elegant result due to [36] on the requirements for shaping functions that lead to unbiased estimates of the value. This will be shown in my thesis. The idea of a shaping function as an "approximate" path integral, based on some heuristic (which can of course also be a dynamic part of the overall integration/optimization processes), can be used immediately to attempt to lower the variance of our estimates by biasing sampling. Recent work by [41] and [29] has also suggested importance sampling. Their work, as it restricts itself to the model-free reinforcement learning scenario, uses importance sampling to re-use samples on multiple controllers. Here instead, we can directly affect the sampling outcome, and will use importance sampling in its more traditional role to lower the variance of an integral update. This approach is simple in that it requires only that we change the one-step forward sampling distribution to sample from the product of our heuristic and the probability and then adjust our estimates. The technology to do this is provided by the likelihood ratio method.

3.1.2 Likelihood Ratio Method of Derivatives The likelihood ratio method of derivative computation [7] is the primary method used in reinforcement learning. It begins with the simple observation that:

$$\nabla_\theta \sum_\Xi P(\xi)R(\xi) = \sum_\Xi P(\xi)R(\xi)\frac{\nabla_\theta P(\xi)}{P(\xi)} \tag{2}$$

and thus that given samples from $P(\xi)$, ξ_i, the estimator

$$\nabla_\theta J = \sum_i R(\xi_i)\frac{\nabla_\theta P(\xi_i)}{P(\xi_i)}$$

is an unbiased estimate of the gradient of the path integral value with respect to the parameters θ. (Assuming that all the derivative and ratio terms exist.) The method known as REINFORCE [5] uses exactly this estimator to compute the derivatives it uses to optimize policies. In a beautiful paper, Baxter et al. [7] review the method and give a rigorous and elegant extension to the average reward case. Fortunately for online-reinforcement learning algorithms, it is assumed easy to sample from $P(\xi)$ and the likelihood ratio can be computed knowing only the likelihood ratio for control probabilities. [7] Of course, with a model we can sample other ways. Suppose we sample at each time step from $Q(x'|x,u)$ instead of the standard $P(x'|x,u)$. Then we need only substitute $Q(\xi)$ for $P(\xi)$ (leaving the derivative alone) in equation (2) to get a different unbiased estimate. Equivalently, we may take samples in REINFORCE and multiply them by the ratio $P(\xi)/Q(\xi)$. One approach to generating samples is to sample at each time step from a substitute distribution. With an underlying MDP structure and a memoryless policy given by μ this ratio is simply the product of the ratio of each transition in the true and substitute simulative distributions:

$$\frac{\prod_t P(x_{t+1}|x_t,u_t)\mu(u_t|y_t)}{\prod_t Q(x_{t+1}|x_t,u_t)\mu(u_t|y_t)} = \prod_t \frac{P(x_{t+1}|x_t,u_t)}{Q(x_{t+1}|x_t,u_t)}$$

One approach to changing our simulative distribution is by "shaping" each transition individually. We may "shape" our sampling distribution by any function $s(x_i) > 0$ defined on the state space and guarantee convergence of the sampling algorithm. By careful application of the likelihood ratio approach, we arrive at the following algorithm (a special case of general path importance sampling) to compute gradients using the multi-step weighted sampling:

Algorithm 1 (Path Importance Sampling) *Takes a shaping function $s(i) > 0$, a parameterized policy $\mu(\theta)$, $\theta \in R^k$, and a POMDP. Returns an estimate of the gradient from 1 trajectory with an importance weight.*

```
1. $a_0 = 0$; $\delta_0 = 0$(vectors of length k)
2. $w_0 = 0$; (scalar importance weight) $t = 0$
3. Do:
    (a) Sample control from probabilistic policy $\mu$
    (b) Sample $x_{t+1}$ from $\frac{p(x_{t+1}|x_t)s(x_{t+1})}{Z_{t+1}}$
    (c) Sample $y_{t+1}$ from observation distribution
```

```
        (d) Generate $r(x_{t+1})$
        (e) $w_{t+1} = w_t \frac{Z_{t+1}}{s(x_{t+1})}$
        (f) $a_{t+1} = a_t + \frac{\nabla_\theta \mu(u_t, y_t; \theta)}{\mu(u_t, y_t; \theta)}$
        (g) $\delta_{t+1} = r(x_{t+1}) a_{t+1} + \delta_t$
        (h) $t = t + 1$

    4. While($x_t$ is not terminal state)
    5. Return gradient estimate $\delta_t$ and weight $w_t$
```

■

The algorithm for generating weighted gradients above is very simple. Although it provides the flavor of algorithms I wish to consider, and space does not permit more detailed description, the above algorithm will suffer from random-walk behavior on importance weights. This particular problem is the bane of sequential importance samplers. Part of the thesis research will be in developing more sophisticated samplers such as Sampling Importance-Resampling and Markov Chain Monte Carlo path samplers.

3.2. Expectation Propagation

The method discussed above attempts to improve upon naive sampling by applying more clever strategies to gain the required samples. A different tack is to apply a deterministic scheme to the integration problem.

A recent advance in estimation was the development of the very general expectation-propagation algorithm. This algorithm extends the celebrated belief-propagation algorithm of Pearl (see also [43]) to general Bayesian inference with approximations in the exponential family. Although first viewed as a rather ad-hoc approach to inference (in non-tree structured graphs) loopy belief propagation attracted wide-spread interest due to coding breakthroughs in the information theory community. It was discovered that Turbo-codes [9] and Low-Density Parity Check Codes [26] [16] achieved decoding bit-error rates very close to Shannon's bound on optimal performance using an iterative decoding equivalent to BP. This lead to a great deal of theoretical attention to the techniques which resulted in more general algorithms, including EP, and deep insights into the relation between BP and approximate integration techniques pioneered in the physics community [52]. The thesis will pursue these connections as they relate to path-integration. Below I discuss in detail Belief Propagation (BP) and extensions to compute derivatives with respect to controllers. In section (3.3.3) I discuss related techniques that I will developed in the thesis. Expectation-Propagation is best introduced by first discussing the Assumed Density Filter [27]. ADF attempts to do sub-optimal filtering by propagating an approximation to the true density in an online system. For example, a non-linear controlled system with a Gaussian distribution over states will immediately evolve in time to a non-Gaussian distribution. Generally, it becomes very difficult to maintain this complex belief state. ADF proposes that at each time step we instead maintain a distribution from a convenient approximating family of distributions, and when propagating forward in time we project back to this family to maintain this simplicity. A natural metric on beliefs is the relative-entropy or KL-divergence between distributions, [11]:

$$D(p||q) = \sum_{\Xi} p(\xi) \log \frac{p(\xi)}{q(\xi)}$$

and thus the canonical version of the ADF projects the resulting belief state at each time step to the distribution in its approximating family that minimizes this divergence. Besides the obvious applications in filtering, ADF has been proposed for numerous less obvious applications including classification, where it forms an online algorithm to compute the posterior distribution of allowed classifers. [30] ADF, as it is clearly sensitive to the ordering of data-points (certainly odd in the classification example), can make poor approximations early in the filtering that are not easily corrected later.

Minka [30] made a doubly interesting realization: first, that ADF within the exponential family can be modified to correct these approximation errors due to ordering, and second that this approximation is a powerful extension to a number of existing algorithms, most notably BP. The key is to note that when performing ADF within the exponential family, there is a relationship between the prior and posterior belief states due to the approximate updates. Instead of treating ADF as performing an exact update and then projecting, equivalently it may be viewed as approximating evidence and then performing an exact update using this evidence. This is clear since we can define the effective update as simply the ratio of the posterior and prior:

$$\tilde{t}_i(\xi) = p'(\xi)/p(\xi)$$

The effective update is also in the exponential family (of unnormalized distributions). With this realization we can take our approximate posterior made up of our original prior multiplied by these approximations:

$$q(\xi) = p(\xi) \prod_i \tilde{t}_i(\xi)$$

and refine it by removing one approximate term, and computing a new approximation, using the exact evidence t_i (here the conditional probability tables) for it in the same way as ADF. We may iteratively do this for all the approximations in turn until we converge on an approximate posterior.

We wish now to extend EP to the path integration problem. Two useful extensions are in order: first, as in path importance sampling, we would like to encourage our approximation to better capture higher cost regions; and second, we would like to compute derivatives with respect to parameters of the policy. Fortunately, both of these extensions can be made to the EP algorithm.

Although a full derivation of the iterative derivative computation will be saved for the thesis document, the idea is simple enough to convey. For concreteness, consider the case of a discrete Bayesian network describing the time evolution of a controlled system. It could be a DBN (Dynamic Bayes Net) describing a POMDP or one with more complicated links that violate the Markov structure. Either way, we have a standard Bayes net sliced into time-slices with state-variables and rewards and controls for those state variables. The true posterior distribution over paths is given by the network structure and conditional probability tables as:

$$p(\xi) = \prod_{t,k} p(x_t^k | par(x_t^k)) \tag{3}$$

Consider rewards that are additive in time as in the MDP case and that are additive across nodes i describing state-variables in the network:

$$R(\xi) = \sum_i \sum_t r_i(x_i^t)$$

In the standard version of belief propagation, we approximate the complete posterior path probability by a completely factored approximate distribution:

$$q(\xi) = \prod_{i,t} q_i^t(x)$$

To simplify notation, indices for state variables will implicitly include time and will run over k. Term approximations t^i will be index by i. The factorization of $q(\xi)$ guarantees that the $\tilde{t}^i$ term approximations will also factorize:

$$\tilde{t}_i = \frac{\Gamma(\frac{q(\xi)p(x_i|par(x_i))}{Z})}{q(\xi)}$$

where Γ is the ADF approximation operator onto the factorized family. That is, we may right each approximate update as a product of updates on each node in the graphical model:

$$\tilde{t}_i = \prod_k \tilde{t}_i^k$$

In particular, $\tilde{t}_i^k$ will only be different than 1 given the standard ADF projection operator when k is in the conditional probability table as either a child or parent variable. Further, it may be easily checked that canonical ADF projector simply maintains marginal probabilities over all the discrete nodes.

Clearly, the derivative of the cost function with respect to θ is simply $\sum_\xi r(\xi)\partial_\theta p(\xi)$, at least in the case considered in this document, where $r(\xi)$ is independent of θ. So we must compute derivatives of our approximation distribution

$$q^k \propto p^k \prod_i \tilde{t}_i^k \tag{4}$$

To compute derivatives of the approximations, we need only compute all the derivatives of the term approximations. From the condition that ADF simply maintains marginals, each derivative approximation:

$$\frac{\partial \tilde{t}_i^k(x_k)}{\partial \theta} = \sum_{\xi : x_k} \frac{\partial t_i(\xi)}{\partial \theta} \prod_{j \neq k} q_j(x_j) + \sum_{\xi : x_k} t_i(\xi) \frac{\partial \prod_{j \neq k} q_j(x_j)}{\partial \theta} \tag{5}$$

consists of the derivatives of the conditional probability tables and terms related to the derivatives of other nodes in the network. We now have a set of fixed point equations for the derivatives. On top of the existing message passing of BP, we can add a message passing for derivatives that stores and propagates these approximate derivatives and enforces the fixed point equations. Full details are reserved for greater space.

In the thesis, I will also present new ADF projection operators that better approximate the path integral. The standard approximation uses relative entropy as its optimization criteria. Instead, we would like to weigh the projection operator to focus accuracy of the approximation on larger contributions to the integral.

It is worth noting that both of these extensions may be of independent interest and have uses outside the realm of control or even decision theory. For example, estimating the derivative of the evidence with respect to parameters of the model would facilitate model selection. Further, in the thesis I will show how this algorithm can be used to compute the kind of derivatives and probabilities needed by the likelihood ratio method for multi-robot problems with exponentially large action spaces while still respecting a required communication structure.

3.3. Other integrators

The bulk of the research I wish to conduct for my thesis is related to the integrators discussed above and other extensions. This field of research seems quite wide open, with great promise for a variety of methods. I will briefly discuss some other methods that suggest themselves and how they may be extended and applied.

3.3.1 Sequential Filtering Research in sequential filtering has developed a number of clever techniques that maybe of use in control. One notable algorithm is the Unscented Kalman Filter [23] (UKF). The UKF is an ADF type algorithm on Gaussians with a clever propagation and projection step. In the UKF, one takes samples along the eigenvectors of the covariance matrix at a distance of 1 standard deviation. These samples are propagated forward using the system dynamics and then used to compute a new Gaussian density by preserving the moments of these samples. Unlike the Extended Kalman Filter, the UKF doesn't require analytic computation of the Jacobian, and tends to be more robust to heavy-tailed distributions over state. In the thesis I would like to combine the UKF with the iterative refinement idea of EP to gain a powerful integrator for continuous problems. This approach too may be useful outside control. For example it provides a new approach to combining online and batch estimation for the Simultaneous Localization and Mapping problem as in [12].

3.3.2 Density Estimation Next, I would like to use non-parametric density estimators as a technique to compute path-integrals. Density estimation for policy evaluation was first considered in [38] and appears to be a technique with much promise, as it may allow us to easily trade bias for lower variance. I hope to improve on the results there by emphasizing trajectories and doing density estimates over these trajectories.

3.3.3 Field-Theoretic Integration In work closely related to belief propagation, a large number of techniques from field-theory and statistical physics have already been brought to bear on inference problems. For example, excellent results have been obtained using variational (particularly structured mean-field) [21] methods, and a series of improvements on this technique including variations on the Thouless-Andersen-Palmer (TAP) approximation, Kikuchi and Bethe approximations, and the cavity, replica and saddle-point methods have all been applied to graphical model inference. (A summary of much of this research is available in [39]). Further, very recent research has focused on the renormalization group as a method to recursively solve large problems [52] [3]. There is a great deal of promise in these techniques and many are closely related to the Expectation-Propagation algorithm discussed above. They are interesting insofar as they may provide deeper insight into the path-integration problem as well as provide new algorithms with better convergence and/or convexity properties. The thesis will address some of these techniques as they prove useful in control.

3.3.4 Combining Integrators Finally, one of the most useful contributions may be the combination of the techniques discussed here and the better known approachs in control. For example, in some problems we will be able to handle some computations exactly and thus reduce the variance of MC estimators (given the unhelpful name ”Rao-Blackwellization” in statistics). In others, some pieces of a larger problem may be amenable to near exact computation, by EP, others amenable to value function approximation, while some parts remain best handled by brute force Monte-Carlo. Engineering good ways to combine these methods will almost certainly be critical to large scale application of stochastic control and planing.

4. Novel Models for Uncertain Systems and Robust Control

Applications to real robotic systems, including Carnegie Mellon’s autonomous helicopter, motivated the development of a new class of models. In these applications it is valuable to develop techniques to handle model uncertainty and under-modeling.

Under-modeling is a certainty; compromise is an inevitable consequence of the need for computationally tractable models– both learned and hand-designed. Ensuring that control algorithms aren’t brittle with respect to these unmodeled dynamics is of paramount importance in model-based reinforcement learning and planning problems. In the first section, I present a slight extension of the path integration approach that enables the development of controllers robust to non-Markovian disturbances.

Even when a model is able to accurately capture the dynamics of a system, and the non-

Markovian effects may be neglible, model-based reinforcement learning must always deal with the limits of available data. In large state-spaces, we constantly face a data-sparsity problem in trying to build a model. The certainty-equivalent approach can fail badly as it assigns complete confidence even when the data is arbitrarily sparse. To be Bayesian about our uncertainty, we should explicitly model our uncertainty about dynamics and do well on average with respect to that uncertainty. In the second section, we consider an extension of the MDP in which we try to minimize cost over a distribution of models.

4.1. Stochastic Robustness

A problem that is of fundamental interest in planning and control problems is determining the robustness of a plan or policy to modeling errors. It has long been recognized in the control community that controllers developed in the framework of stochastic optimal control may exhibit unsatisfactory online performance due to poor robustness to small modeling errors. More recently, this problem has been recognized in the reinforcement learning community as well, and algorithms have been suggested to deal with it. Minimizing the risk and impact of brittle controllers is of some import in model-based reinforcement learning solutions, particularly ones where failure has significant consequences [2].

One basic approach is to abandon the framework of stochastic control entirely. [19] and [33] adopt this approach, replacing a probabilistic model of interactions with a deterministic, worst case evaluation criterion. [19]'s technique (mini-max Q-learning) assumes that at each step the worst possible result happens to an agent. As might be imagined, this can lead to rather conservative estimates of the value of a policy, particularly in the instances where a stochastic model is an accurate representation. In the worst case- a truly stochastic model where there is some small probability of catastrophe at each state, this algorithm evaluates all policies as equally abysmal performers.

Much work in the control community has been directed towards this problem, and the formulation known as H_∞ robust control has been a research focus for decades. The efficient computational solution to the linear problem has attracted a great deal of interest among practitioners. H_∞, generalized to non-linear problems is closely related to the mini-max criterion described above as in this framework the controller is pitted against a *disturber* that can inject an L_2 bounded disturbance into the feedback loop. The controller seeks to maintain the stability of the system by attenuating this disturbance. The relation between disturbances in the H_∞ formulation and model error can be attributed to results like the *small-gain theorem* [49]. The relationship is particularly sharp in the linear case, where the disturber can be identified with a norm-bounded (in the Hardy-space, H_∞, hence the name) linear perturbation. In the context of finite-state machines (deterministic MDP's), H_∞ reduces to the mini-max framework described above with the addition of costs incurred against the disturber for the "energy" required for applying each disturbance. [33] shows how H_∞ might be phrased in terms of reinforcement learning.

Stochastic models were developed to model situations where state transitions happen due to events that are largely unobservable and uncontrollable. While H_∞ is doubtless a useful scheme for control, there are many applications of practical interest where a stochastic model remains an excellent approximation of real system behavior, and a worst case deterministic model has both very poor fidelity with the real system and leads to controllers with performance so conservative as to have little practical value. Yet we still wish to ensure that our policies remain robust to errors made in modeling the stochastic process. A fundamentally different approach has been taken in *risk-sensitive* optimal control. In the risk sensitive framework, the basic structure of a stochastic model is retained, but the risk-neutral additive cost function is replaced with a cost function that emphasizes variance in cost occurred during control– in particular, risk-sensitive controllers will prefer a deterministic reward r to a lottery l with $E[l] = r$. This seems a natural preference– at least for human decision makers. Risk sensitive criteria can also be related to a type of model uncertainty; [14] connects risk-sensitive control, where the value is defined as the time-average expectation of an exponential sum of costs, to a dynamic game, similar to the H_∞ problem, where the disturber influences next state transitions distributions, but pays a penalty for the relative entropy incurred at each step between the nominal model and the disturber's preferred next state distribution. The risk sensitive criterion thus provides a link to our desiderata of robustness, but lacks strucuture: deviations in the transition probabilities at every state are considered equally and no bound is imposed to prevent possibly arbitrarily large deviations from a nominal model by the disturber.

4.2. Sets of Path Distributions

I propose to address the stochastic robustness problem more directly. Uncertainty will be described in terms of a set (henceforth the *uncertainty set*) of possible probability distributions over paths. The problem will be to find a stationary, Markovian policy (the set of which is denoted Π) that performs optimally in the sense that it maximizes the expectation of reward over all possible path distributions. As in standard MDPs, we will consider problems with a (finite) state space by $\mathcal{X}$ and the (finite) space of controls by $\mathcal{U}$. The discount factor is indicated by γ, the reinforcement function over states by $R(x)$, the uncertainty set of distributions of trajectories by $\mathcal{P}$, and the set of possible next state distributions drawn from $\mathcal{P}$ corresponding to a state i and control u as $\mathcal{P}_i^u$. (i.e. all possible marginal distributions starting at i with control u)

To ensure robustness we wish to find controllers that perform well on *all* models in the uncertainty set. Specifically, envision a static game in which a stationary Markovian controller is chosen, and then a model is chosen from the uncertainty set so as to minimize the expected reinforcement recieved by the controller. That is, the game's value (to the controller) is defined as:

$$\min_{p \in \mathcal{P}} \max_{\pi \in \Pi} E_{p,\pi}[\sum_t \gamma^t R(X_t)] \tag{6}$$

4.2.1 General Sets It is unfortunately the case that finding policies that are guaranteed to be good across unrestricted sets of models is computationally challenging. A stronger claim is also true: Although we have allowed arbitrarily complicated non-Markovian models into our uncertainty sets to allow for under-modeling, the computational difficulty is not mitigated by restricting the uncertainty set to only contain Markov Decision Processes. A reduction similar to Littman's proof [24] of the hardness of finding optimal memoryless policies in POMDPs proves the following result:

Proposition 1 *Finding the stationary memoryless policy that maximizes the least expected reward over an uncertainty set of Markovian Decision Problems is NP-hard.* ■

4.2.2 Convex Sets I will now consider more restricted uncertainty sets that are both broad enough to be quite interesting with regard to applications, yet restrictive enough to admit tractable solutions. Two essential conditions must be met. First, we require that the uncertainty *factor* by state/action pair– that is, for each state-action pair the next-state distribution can be chosen by the minimizing player independently of the next state-distribution chosen elsewhere in the state space. Clearly the uncertainty set can contain distributions where this doesn't hold as well as the ones where it does. Secondly we require that the possible marginal probability distributions $P(j|i,u)$ form a convex set. The uncertainty set of path distributions will include an uncountable set of MDPs, as well as non-MDPs that meander around inside the limits set by the marginal distributions. In terms of just the MDP subset of the path distributions, the definition can be cleanly stated in terms of a kind of convexity of the next-state transition kernels:

Definition 1 *We call an uncertainty set of MDPs* convex *if for every action u, every state i, and any two transitions kernels defining the distribution over next states T^u_i and S^u_i, all next-state distributions "on the line" between the S and T are included. That is, all transition functions of the form*

$$V^u_i = \alpha(i,a)T^u_i + (1-\alpha(i,a))S^u_i$$

for any function $\alpha(i,a)$ *taking its values in the range* $[0,1]$*, are also in the uncertainty set.*

To support the claim that *convex uncertainty sets* are interesting, I provide some examples. First, there are sets of MDPs such as:

Example 1 *The class of uncertainty sets known as* interval-MDPs *[51] where every element of the transition matrix has bound constraints such as $\zeta^u_{ij} \leq P^u_{ij} \leq \psi^u_{ij}$ is a convex uncertainty set.*

Example 2 *The class of uncertainty sets where each row of the the transition matrix is constrained to lie in some* relative-entropy *ball around a nominal distribution ζ^u_i, $\{P^u_i | D(P^u_i|\zeta^u_i) \leq \epsilon^u_i\}$, describes a convex uncertainty set.*

Example 3 *In the mini-max model next states are chosen independently and deterministically for each state-action pair. The uncertainty set consisting of for each state-action pair the convex hull of the possible next states in the probability simplex constitutes a convex uncertainty set.*

Some interesting non-MDP models fall within the convex undertainty set definition as well. Consider some notions of stochastic processes that behave as if they were "nearly" Markovian, and then discuss algorithms designed to behave robustly to this form of perturbation.

Definition 2 *Two controlled stochastic processes, P and Q are ϵ-close in variation if $P(x'|x,u) - \epsilon \leq Q(x'|x,u) \leq P(x'|x,u) + \epsilon$ holds for all x, x', u and all possible trajectories of the process. Controlled stationary stochastic process P is ϵ-close in relative entropy to processes Q if for all possible trajectories of each process*

$$sup_{x,x',u} D(P(x'|x,u)|Q(x'|x,u)) \leq \epsilon$$

Definition 3 *A controlled stochastic processes X is* boundedly non-markovian in variation (relative entropy), *with bound ϵ if there exists a Markov process with transition matrix T such that T and X are ϵ close in variation (relative entropy). T is denoted the nominal model for X.*

Example 4 *The set of all $epsilon$-boundedly non-markovian decision problems relative to a nominal model T forms a convex uncertainty set.*

First observe that MDP solutions possess moderate inherent robustness to boundedly non-Markovian behavior of a process:

Theorem 1 *Solving for the optimal policy in a nominal Markovian decision problem T, corresponding to a boundedly non-markovian decision problem (with bound ϵ in either variation or relative entropy) induces error over the optimal policy that is polynomially bounded in ϵ.*■

4.3. Solution Algorithms

The factored state assumption condition that suggests the introduction of a stochastic dynamic game with turns between a controller that chooses an action u at each time step, and a disturber who counters with the next step distribution $P(j|i',u)$ from a set of allowable next state distributions $\mathcal{P}_i^u$.

Note that the dynamic game described above is a zero-sum one of complete information. It follows from well know results in game theory [6] that the following holds:

Lemma 1 *If $\mathcal{P}_x^u$ defines a compact region of the probability simplex for every x and u, then there exists for both the controller and disturber Markovian, stationary, and deterministic optimal strategies.* ■

Note that value-iteration in the dynamic game implicitly gives a policy for both the disturber and the controller by look-ahead in the one-step game. Surprisingly, although the dynamic game was introduced as a relaxation of the game defined in equation (6), as it appears to give much more power to the controller and disturber, the above result implies that under the further assumption that $\mathcal{P}$ factors by state and action, the dynamic game solution is *also* a solution to the static game (6) of uncertain models described above:

Theorem 2 *If $\mathcal{P}$ is a compact and convex uncertainty set of distributions, then the optimal disturbance is a stationary, Markovian distribution over next states and can be identified with an element of the subset of MDPs in $\mathcal{P}$. Robust value-iteration in a compact convex uncertainty converges to the value function of the optimal policy with respect to worst model in $\mathcal{P}$.*

I present the algorithm:

Algorithm 2 (Robust Value Iteration)

1. *Initialize V to the zero vector the size of the state space.*
2. *Repeat until V converges:*
3. *For all states i and controls a, assign to matrix Q_{min} the solution to the optimization problem:*

$$Q_{min(i,a)} = \min_{p \in \mathcal{P}_i^u} E_p[V + R(i)]$$

4. *Update V by maximizing over the control set:*

$$V(i) = \max_{a \in \mathcal{A}} Q_{min(i,u)}$$

■

This algorithm can also be easily extended to run asynchronously and online. It is not clear however, that Algorithm (2) leads to a computable solution, as Step 3 requires a minimization over an uncountable set of probability distributions. Perhaps surprisingly it is not only computable but tractable:

Corollary 1 *Finding the (near)-optimal policy to guarantee performance (within a given ϵ) in a compact and convex uncertainty set of MDPs is a polynomial time problem, and Algorithm (2) solves the problem in polynomial time.*
Proof: (Sketch) Note that the minimization required in the algorithm is a convex program: The expectation is linear in p and thus forms a linear objective function subject to convex constraints. It is known that convex programs can be solved in polynomial-time (in the number of states) by interior point (e.g. ellipsoidal algorithm) methods. Further, each epoch of value-iteration reduces the error geometrically, so that in time logarithmic in ϵ, the maximum allowed sub-optimality, an approximately optimal policy can be computed. ■

In sharp contrast to the general problem, convex uncertainty sets lead to a tractable solution to the uncertain model control problem. The algorithm presented, as it requires convex programs to be solved at each step can be quite expensive in practice, despite the polynomial time guarantees. However, the algorithm naturally takes advantage of structure in the uncertainty set. In the case of variation bounds on the probabilities, the convex program reduces to a linear program, and so for small structured uncertainty sets, the LP can reduce to a simple minimization over a finite set of possible disturbances corresponding to the vertices of the constraint set.

4.4. Connections to H_∞

As discussed above, methods other than stochastic optimal control with respect to the nominal model exist for mitigating the impact of non-markovian behavior. Of particular interest is the H_∞ paradigm, where energy costs are associated with disturbances injected at each time step.

The simple modification of step (3) of Algorithm (2) to include energy costs as follows:

$$Q_{min(i,a)} = \min_{p \in \mathcal{P}_i^u} E_p[V + R(i) + \lambda S(i,j)]$$

where $S(i,j)$ is a non-negative cost associated with each transition (from i to j), representing energy injected by the disturber to induce unfavorable transitions and λ is the scalar H_∞ coefficient, generalizes the standard discrete state, discounted H_∞ problem. It is easy to recover the standard formulation by relaxing all of the next-state distribution constraints to include the entire simplex, and ensuring that for every state i there exists a j so $S(i,j) = 0$.

4.5. Experiments with Stochastic Robustness

4.5.1 Path Planning Robot path planning has been the subject of a great deal of research effort, but the vast majority of it concentrates on static obstacles. A significant difficulty in planning with dynamic obstacles is the difficulty of modeling such objects– the dynamics of a human obstructing the robot's path are stochastic and possibly very high dimensional.

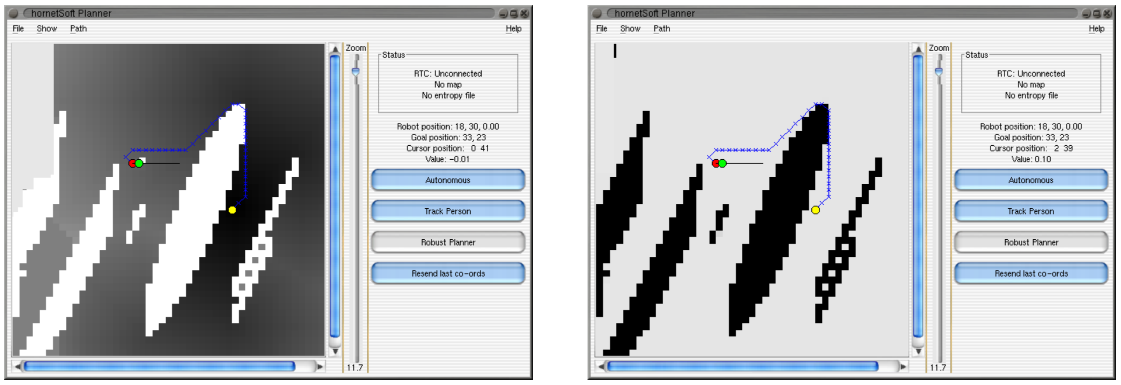
It is possible to use the framework here to develop controllers robust to a wide variety of dynamic objects. Specifically, the dynamic obstacle is modeled as a "lazy" random walk with unknown and time-varying drift. In the eight connected grid used for planning, the dynamic obstacle is subject only to the linear constraints:

$$Pr(\textit{Obstacle doesn't move}) \geq .25$$

This gives a vast space of possible dynamics (transition matrices) for the object whose details we leave unspecified. In this model the dynamic obstacle is thought of as inadvertently adversarial (perhaps not an unrealistic assumption for some humans interacting with the robot), and we require path planning to remain robust to all such obstacles.

In many situations paths resulting from the uncertain MDP solutions result in paths that show greater deference to the impedance caused by the dynamic obstacle than solutions that treat the object as static. Figure (1) illustrates the robot circuiting the obstacle, and the resulting value function. Lighter regions indicated regions of higher cost. As the resulting plans are feedback strategies, the illustrations depict the plan resulting from the object staying still. (The computed strategy, of course, does not.) A conventional planner skirts the obstacle to minimize cost.

In the Figure (2) I compare solutions derived by the standard planner and the robust planner, noting that to avoid interference from the dynamic obstacle, the robust controller takes a longer and very different path to the goal, despite the existence of a feasible (although potentially quite difficult) path between the dynamic obstacle and the other impediments in the environment.



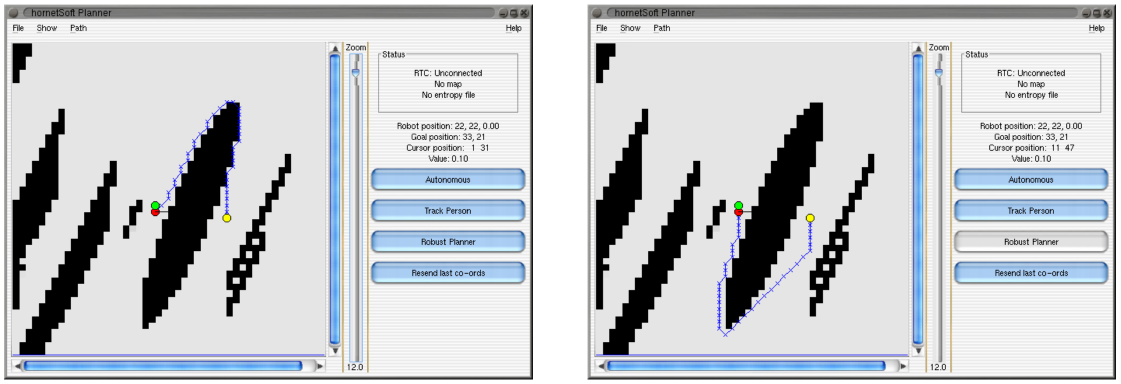
RTC: Unconnected
No map
No entropy file

Robot position: 18, 30, 0.00
Goal position: 33, 23
Cursor position: 0 41
Value: −0.01

RTC: Unconnected
No map
No entropy file

Robot position: 18, 30, 0.00
Goal position: 33, 23
Cursor position: 2 39
Value: 0.10

Figure 1: These figures illustrate a path planned by the robot (dark leftmost circle) giving a dynamic obstacle (lighter circle adjacent to the robot) a wide berth on it's route to the goal (light circle on far right). The figure on right illustrates the value function compute by the robust dynamic programming algorithm while the figure on the right illustrates the path taken.



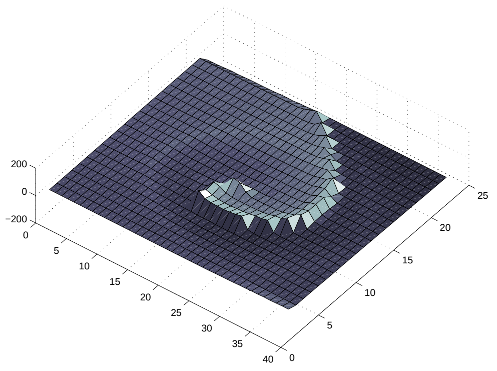
RTC: Unconnected
No map
No entropy file

Robot position: 22, 22, 0.00
Goal position: 33, 21
Cursor position: 1 31
Value: 0.10

RTC: Unconnected
No map
No entropy file

Robot position: 22, 22, 0.00
Goal position: 33, 21
Cursor position: 11 47
Value: 0.10

Figure 2: The left figure shows the plan generated by the conventional dynamic programmer. On the right the uncertain MDP solution leads to a path (from the dark circle robot to the light circle goal) that makes a greater effort to avoid the dynamic obstacle.



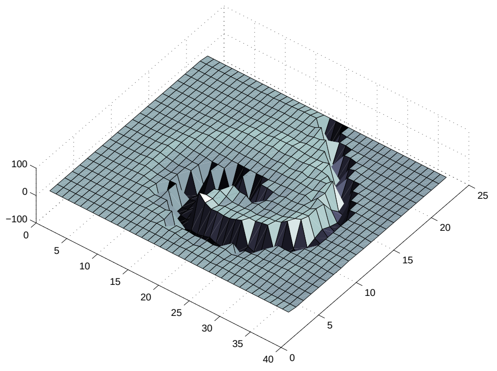


Figure 3: A graphical depiction over the coarsely discretized state-space of the relative performances of the robust and standard controllers. The left graph illustrates the degree to which the standard MDP algorithm underestimates the true value of each state, while the right graph illustrates the relative performance of the two controllers in simulations. Positive values indicated the robust solution out-performed the standard one.

4.5.2 The Mountain-Car POMDP It is also interesting to look at discretized and hidden state. In particular, consider a variation on the well-studied "Mountain-Car" problem, where unlike the standard formulation, only an approximate, discrete representation of the state is available for control.

The dynamics used are the standard ones given in [47], but complete state observability is replaced by discrete valued sensors that output the car position to within .085 position units ($\frac{1}{20}$ of the cars position range) and .007 velocity units ($\frac{1}{20}$ of the maximum speed). Two approaches are presented to solving the problem. First the state uncertainty is treated by approximating the problem with a Markovian one. Assuming that for a given sensor measurements all states that result in this sensor measurement are equally likely, I compute transition probabilities by sampling one step forward from a uniform distribution over each sensor grid cell. Standard dynamic program is run on the resulting discrete MDP. Next, I use the robust value iteration described above to model the possible error introduced by assuming the problem is a discrete, fully-observed MDP.

Both methods are able to find satisficing controllers, however, the robust algorithm's value function serves as a lower bound on the actual cost when the controller is run on a simulation of the mountain car dynamics. The MDP solution often significantly under-estimates the actual cost of a state. (Figure 4.5.2) The controller developed using the robust value iteration algorithm is quite conservative, preferring multiple swing-ups to ensure success. In a number of places it still out-performs the standard controller. (Figure 4.5.2).

It is interesting to observe that the sharp under-estimates of the true cost of a policy in the MDP do *not* vanish with increased resolution of the discretization. (See Figure(4.5.2)). This is a very general phenomenon that occurs along switches in the optimal policy, so that given a δ-fine discretization, one can expect these kind of inaccurate value estimates to occur on $O(1/\delta)$ of the state space. MDP solutions are helped by noise in this case as it smoothes the dynamics, making the uncertainty in the transitions due to discretization relatively small.

Although space does not permit details of experiments, it is worth noting that the framework of stochastic robustness is very natural for model-based reinforcement learning, and enables the

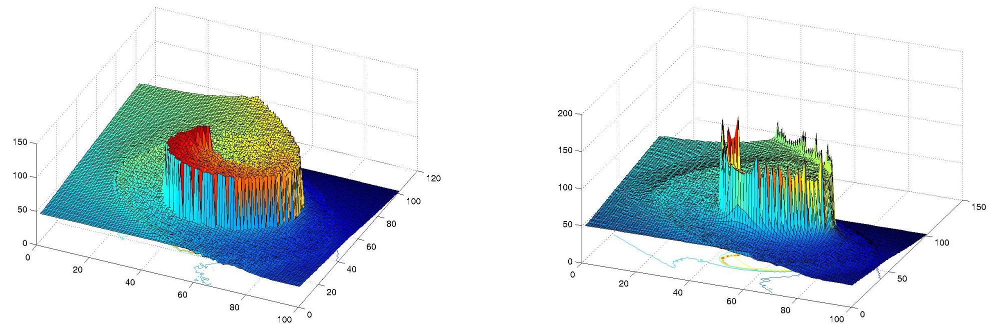


Figure 4: The left figure illustrates the mountain-car value-function computed by Algorithm (2) on a fine discretization of state-space. On the right the we see the simulated performance of the MDP solution. Note the difference in vertical scale.

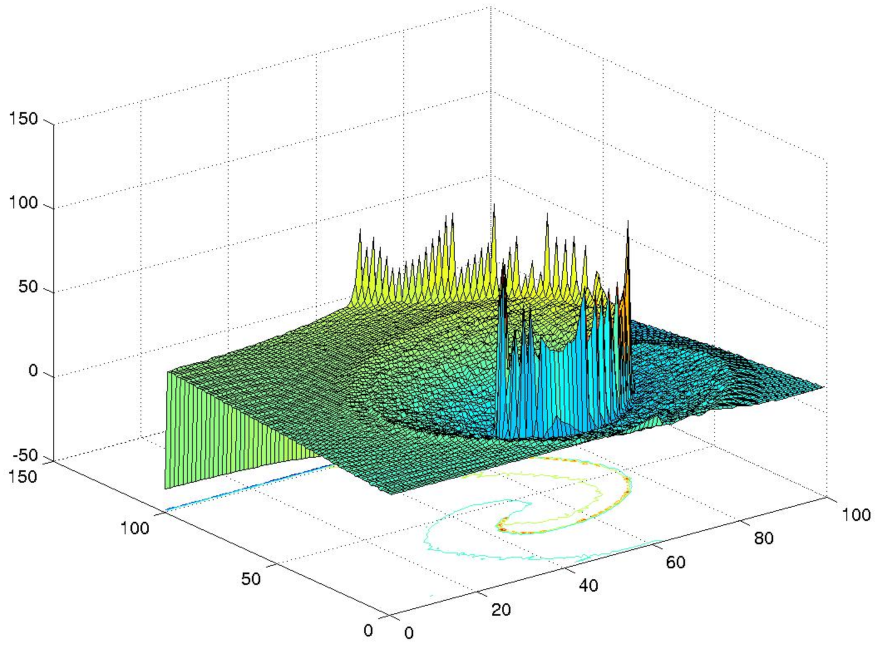


Figure 5: Error in the MDP cost estimate at a fine discretization. Note the sharp underestimates that occur along the decision boundary.

learning of controllers that can guarantee (with high probability) performance while learning, ensuring robustness that the certainty-equivalent approach cannot.

4.6. Uncertain MDPs

In this section I outline work done on an extension of MDP models that arises in the context of model-based reinforcement learning. 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. It is our contention that these techniques face serious difficulties in the application to robotics.

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", [8]). 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.

4.7. 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.

To consider this as a metric on policies, I suggest that policies be ordered by mean trajectory performance, where the expectation is taken with respect to measure P (including Markov noise and *model distribution*). 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. Under this metric, 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.

As in the Stochastic Robustness model, there are times when it is valuable to consider stronger guarantees on performance than simply average-case. We can extend our criteria for safety and robustness criterion is by maximizing the performance on the worst model in a large set of models (large in the sense of measure, e.g. on .95 fraction of the 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 [32].) 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.

One should briefly note that all of the results on the sample complexity of the evaluating a policy class developed in [37] applies equally to our problem of averaging over models. In fact, the clever PEGASUS trick of fixing events in the samples space (equivalently the random number generator) was used throughout my experimental work and is quite effective in reducing the computation needed to optimize the controllers. Despite these guarrantees, it is worth pointing out that the following is true:

Proposition 2 *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 [24] proves the result.

4.8. Sampling Algorithms

Until this point I have deferred the question of sampling from the space of trajectories Ξ. 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. [1] 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

Figure 6: The CMU Yamaha R50 helicopter in autonomous flight.

distribution due to model uncertainty. I 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 hopefully are close approximations to the ideal samples.

Algorithm 3 (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, 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 3 (Sufficiency of 1-step predictive learners) *If model M in algorithm (3) can be recursively updated in the Bayes-optimal way, the trajectories generated by the algorithm (3) are independent samples from the n-step predictive distribution.*■

4.8.1 Helicopter Experiments The research described in the section was pursued in large part to facilitate the development of controllers of physically realized robotic systems– and in particular Carnegie Mellon's autonomous helicopter. 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 I 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.

4.8.2 Dynamics To provide a manageable first goal in applying policy-search to the helicopter, I 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. [28] 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.

4.8.3 Modeling Modeling a dynamical system is always challenging. To learn the dynamics of the helicopter, I chose to implement a LWBR (locally weighted Bayesian regression) state-space model. This is a powerful non-linear learning algorithm that allowed us to capture uncertainty in the dynamics of the helicopter. Details on the learning algorithm are available in [31] and applied to the helicopter in [2].

4.9. Controller design

4.9.1 Controller structure In proposing an initial controller structure, I looked towards simple controllers known to be capable of flying the helicopter. To this end, I proposed a simple, neural-network style structure 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.

4.9.2 Optimization It has previously been demonstrated that the criterion of averaging trajectory costs over noise and models (or rather the approximation of it given in [45]) 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, mostly variants on quadratic penalty functions, each leading to different (although mildly so) controllers. Details are available in [2]. In all cases there was a large penalty (10^6) for state-variables that were outside the

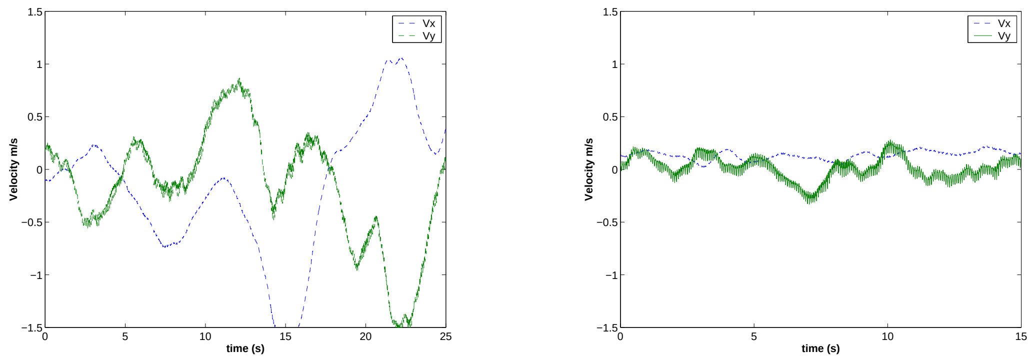


Figure 7: Data logs from the R-50 demonstrating performance hovering under (left) a highly trained pilot's control, and (right) the neural-net controller built by the robust policy search methods.

space of the data we had observed. After establishing the cost criterion, one must consider the task of optimizing the parameters. Trajectories were rolled out using the sampling technique described in algorithm (3) for the LWBR model. Typical policy evaluations were 30 trajectories of horizon length 500 with discount factor $\gamma = .995$.

4.10. Validation

Initial validation experiments were performed on a linear model of the rotorcraft about hover given in [28]. 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.

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.

The helicopter was test flown with the learned controller. 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 (7) 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.

5. Future Applications

This research in integration technology may be applied to a variety of tasks. I am interested in beginning testing on a set of well-studied problems in the AI literature. These include the BATmobile [15] and the SysAdmin multi-agent problem [18].

Next some of the applications will also come from FIRE (Federation of Intelligent Robotic

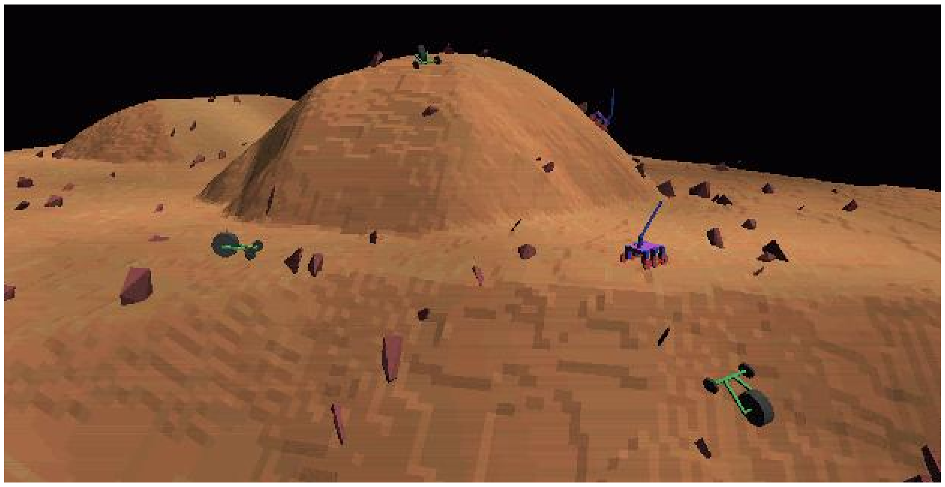


Figure 8: A depiction of the FIRE simulator showing autonomous rovers and the rocks they are investigating.

Explorers), a NASA multi-robot domain. In particular, I will consider the problem of developing controllers that efficiently direct multiple robots in the search for interesting rock-specimens when each individual robot has a significant chance of catastrophically failing. As the initial problem for the system, I will consider a system where there is a team of approximately 10 robots whose goal is to pick up all the rocks (with known locations) in a discretized grid world. Each robot will have a controller with a policy defined in terms of an exponential distribution over local features and features of nearby robots. With probability $1 - \epsilon$ a robot will cease to function at each time step. The goal will be to collect as many rocks as possible before all the robots fail. This problem is interesting as is an NP-hard stochastic control problem with a large space of actions. My preliminary work on the FIRE project has consisted of solving hard inference problems– determining where rocks are likely to be, for example, with a statistical model of rock distributions, and then applying simple planners to multiple agent search. It would be natural to extend the proposed method by considering unknown rock distributions and allowing the robots to maintain certain belief state over that. It is quite likely that multiple techniques will have to be merged here to achieve significant progress. I will also consider continuous and hybrid state problems, such as more sophisticated controllers for the helicopter, where other techniques discussed in this document may prove valuable.

6. Contributions and thesis research

In summary, in my thesis I propose to highlight the connection between path integration and stochastic control. There are three basic components to the research. The first is the insight that may be gained by emphasizing the integration over trajectories aspects of the problem. I hope to clarify existing results in the field and will establish that the connection with the Bellman equation/value function approach to control is one of convex duality. Secondly, I have already developed models, inspired by the focus on trajectories, that have lead to useful results in the field of robust control. The first extended the optimization of the path integral to a adversarial game, while the second treated uncertainty with a distribution over possible models. To complement these models, I developed algorithms that allowed controllers to be built efficiently for them. This led to the first successful reinforcement learning control of a helicopter, as well as the first successful controller for the Robotics Institute's autonomous helicopter project that was not hand-crafted.

Finally, the bulk of future work will focus on the development of approximate integration techniques appropriate to the control domain. Although much research on searching for controllers (policy-search) has focused on the optimization aspect of the problem, there is a great deal of room to adopt new methods of integration. In particular, I will focus on stochastic methods, such as path importance sampling and density estimation, as well as deterministic strategies, such as field theoretic approximation and sequential filtering.

Schedule

As I am suggesting a rather broad topic, I would like to leave myself some time to achieve interesting results. I intended to spend Summer '02 working on the Derivative Propagation and Sequential Filtering approaches on the well-developed problems in the literature mentioned above and in the FIRE domain. In Fall '02 and Spring '03 I will continue this research on FIRE focusing on the multi-agent aspects of this problem. In Spring '03 and Summer '03 I will explore the continuous state and sampling aspects of the problem. In the Fall '03, I will focus on the documentation of this research in the thesis document.

More specifically, my schedule is as follows:

- Summer '02
  - Develop derivative propagation (See section 3.2)
  - Develop sequential filtering algorithm (See section 3.3.1)
  - Apply to initial FIRE problem
- Fall '02
  - Develop muti-agent coordination algorithm using derivative propagation for FIRE domain
  - Apply the two algorithms above to other problems selected from the application section
- Spring '03
  - Develop path importance sampling (See section 3.1)
  - Integrate belief state into FIRE robot policies
  - Develop density-estimation techniques (See section 3.3.2)
- Summer '03

  - Investigate field-theoretic techniques to approximate path integrals (See section 3.3.3)
  - Explore combining various techniques (For example, section 3.3.4)
  - Apply these algorithms to other problems in the application section (5)

- Fall '03
  - Document the work developed in the thesis
  - Defend the thesis

References

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

[2] J. Bagnell and J. Schneider. Autonomous helcopter control by policy-search based reinforcement learning. In *Proceedings of the 2001 IEEE Int. Conference on Robotics and Automation*. IEEE, 2001.

[3] J. Bagnell and J. Schneider. Uncertain markov decision processes. Technical Report CMU-RI-TR-01-26, Carnegie Mellon University, 2001.

[4] L. Baird. Residual Algorithms: Reinforcement Learning with Function Approximation. In *Machine Learning: Proceedings of the Twelfth International Conference*. Morgan Kaufman, 1995.

[5] L. Baird and A. Moore. Gradient descent for general reinforcement learning. In *Neural Information Processing Systems 11*, 1998.

[6] T. Basar and G. J. Olsder. *Dynamic Noncooperative Game Theory*. SIAM, 1995.

[7] J. Baxter, L. Weaver, and P. Bartlett. Direct-gradient-based reinforcement learning i: Gradient estimation algorithms. Technical report, Computer Sciences Lab, Australian National University, 1999.

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

[9] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo codes. In *IEEE Int. Conf. on Communications*, pages 1064–1070, 1993.

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

[11] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1997.

[12] M. Deans. *In preparation*. PhD thesis, Carnegie Mellon University, 2002.

[13] R. Feynman and A. Hibbs. *Quantum Mechanics and Path Integrals*. McGraw-Hill, 1965.

[14] W. H. Fleming and D. Hernandez-Hernandez. Risk-sensitive control of finite state machines on an infinte horizon i. *SIAM Journal of Control and Optimization*, 35, 1997.

[15] J. Forbes, T. Huang, K. Kanazawa, and S. Russell. The BATmobile: Towards a bayesian automated taxi. In *Proceedings of the Fourteenth Interational Conference on Artificial Intelligence*, 1996.

[16] R. Gallagher. Low density parity check codes. *IRE Transactions on Information Theory*, 8(1), 1962.

[17] G. Gordon. Stable function approximation in dynamic programming. In *The 12th International Conference on Machine Learning*, 1995.

[18] C. Guestrin, D. Koller, and R. Parr. Multi-agent planning with factored MDPs. In *Advances in Neural Information Processing Systems (NIPS-14)*, 2002.

[19] M. Heger. Consideration of risk in reinforcement learning. In *International Conference on Machine Learing*, 1994.

[20] Kirill Illinksi. *Physics of Finance: Gauge Modeling in Non-equilibrium Pricing*. Wiley, 2001.

[21] T. Jaakkola. *Variational Methods for Inference and Estimation in Graphical Models*. PhD thesis, Massachusetts Institute of Technology, 1997.

[22] M. Jordan and C. Bishop. *Graphical Models*. In production, 2002.

[23] S. Julier and J Uhlmann. A new approach for filtering non-linear systems. In *In the Proceedings of AeroSense*, 1997.

[24] M. Littman. Memoryless policies: Theoretical limitations and practical results. In *From Animal to Animats 3: Proceedings of the 3rd International Conference on Simulation and Adaptive Behavior*, 1994.

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

[26] D. MacKay. Near Shannon limit performance of Low Density Parity Check codes. *Elect. Letters*, 1996.

[27] P. Maybeck. volume 1, chapter 12. Academin Press, 1986.

[28] 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.

[29] N. Meuleau, L. Peshkin, and Kee-Eung Kim. Exploration in gradient-based reinforcement learning. Technical Report 03, Massachusetts Institite of Techology AI Lab, April 2001.

[30] T. Minka. *A Family of Algorithms for Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, June 2001.

[31] A. Moore. *Efficient Memory-Based Learning for Robot Control*. PhD thesis, University of Cambridge, November 1990.

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

[33] Jun Morimoto and Kenji Doya. Robust reinforcement learning. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 1061–1067. MIT Press, 2001.

[34] K. Murphy. A survey of POMDP solution techniques. Technical report, Berkeley Computer Science, 2000.

[35] R. Neal. Probabilistic inference using markov chain monte carlo methods. Technical Report 1, University of Toronto, January 1993.

[36] A. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, 1999.

[37] A. Ng and M. Jordan. Pegasus: A policy search method for large MDPs and POMDPs. In *Uncertainty in Artificial Intelligence, Proceedings of the Sixteenth Conference*, 2000.

[38] A. Ng, R. Parr, and D. Koller. Policy search by density estimation. In *Advances in Neural Information Processing Systems (NIPS-12)*, 2000.

[39] M. Opper and D. Saad. *Advanced Mean Field Theory*. MIT Press, 2001.

[40] M. Peskin and D. Schroeder. *An Introduction to Quantum Field Theory*. Perseus Books, 1995.

[41] D. Precup, R. Sutton, and S. Singh. Eligibility traces for off-policy policy evaluation. In *Seventeenth Internation Conference on Machine Learning*, 2000.

[42] M. Putterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[43] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–285, 1989.

[44] N. Roy. Personal communications, 2001. Sampling and simulation approaches to control.

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

[46] Steven Shreve. *Stochastic Calculus and Finance*. CMU Lecture Notes, 2001.

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

[48] G. Tesauro. Temporal difference learning and TD-gammon. *Communications of the ACM*, 38:58–68, 1995.

[49] A. J. van der Schaft. *L_2-gain and Passivity techniques in Non-linear Control*. Springer-Verlag, 1999.

[50] M. Wainwright. *Stochastic Processes on Graphs with Cycles*. PhD thesis, Massachusetts Institute of Technology, 2002.

[51] P. Wurman and M. Wellman. Bounded parameter markov decision processes. 1996.

[52] J. Yedida, W. Freeman, and Y. Weiss. Generalized belief propogation. In *Advances in Neural Information Processing Systems (NIPS-13)*, 2001.