# Continuous-State POMDPs with Hybrid Dynamics

**Emma Brunskill, Leslie Kaelbling, Tomas Lozano-Perez, Nicholas Roy**
Computer Science and Artificial Laboratory
Massachusetts Institute of Technology
Cambridge, MA
emma,lpk,tlp,nickroy@csail.mit.edu

**Abstract**

Continuous-state POMDPs provide a natural representation for a variety of tasks, including many in robotics. However, existing continuous-state POMDP approaches are limited by their reliance on a single linear model to represent the world dynamics. We introduce a new switching-state (hybrid) dynamics model that can represent multi-modal state-dependent dynamics. We present a new point-based POMDP planning algorithm for solving continuous-state POMDPs using this dynamics model. We also provide a constrained optimization approach for approximating the value function as a mixture of a bounded number of Gaussians. We present results on a set of example problems and demonstrate that when different degrees of state accuracy are needed to accomplish a task, our hybrid continuous-state approach outperforms a standard discrete state technique.

## 1 Introduction

Partially observable Markov decision processes (POMDPs) (Kaelbling, Littman, & Cassandra 1998) provide a rich framework for describing a number of planning problems that arise in situations with hidden state and stochastic actions. Most prior work has focused on solving POMDPs with discrete states, actions and observations.

However, in many applications, such as navigation or robotic grasping, the world is most naturally represented using continuous states. Though any continuous domain can be described using a sufficiently fine grid, the number of discrete states grows exponentially with the underlying state space dimensionality. Existing discrete state POMDP algorithms can only scale up to the order of a few thousand states, beyond which they become computationally infeasible (Pineau, Gordon, & Thrun 2006; Spaan & Vlassis 2005; Smith & Simmons 2004). Therefore, approaches for dealing efficiently with continuous-state POMDPs are of great interest.

Previous work on planning for continuous-state POMDPs has typically modeled the world dynamics using a single linear Gaussian model[1] to describe the effects of an action (Brooks *et al.* 2006; Porta *et al.* 2006; Thrun 2000). Unfortunately, this model is not powerful enough to represent the multi-modal state-dependent dynamics that arise in a number of problems of interest. Consider a robot hopping across an environment of sand and flat concrete. If the robot is on sand, a single motion will only move it forward a small amount, but on concrete the robot can move forward a large amount. If instead the robot has faulty jump actuators, a single action may jump the robot forward, or fail to move the robot. Though such dynamics are easily represented in discrete-state environments using the standard transition matrices, a single linear Gaussian continuous-state model will be insufficient to adequately model these multi-modal state-dependent dynamics. In this paper we present a hybrid dynamics model for continuous-state POMDPs that can represent a stochastic distribution over a number of different linear dynamic models. This dynamics model can also compactly represent the shared dynamics of many states, in contrast to typical discrete state models which require specifying the dynamics separately for each state.

We develop a new point-based approximation algorithm for solving these hybrid-dynamics POMDP planning problems that builds on Porta et al.'s continuous-state point-based approach (Porta *et al.* 2006). A second contribution of our paper is the use of constrained optimization to approximate the value function in order to ensure computational tractability. Constrained optimization provides a principled way to balance representational accuracy and computational efficiency when computing the approximation.

We present experimental results on a set of small problems to illustrate how the representational power of the hybrid dynamics model allows us to address problems not previously solvable by existing continuous-state approaches. Finally, we also demonstrate how our hybrid approach can outperform discrete state approaches when various levels of representational granularity are needed to find a good policy.

## 2 POMDPs

Partially observable Markov decision processes (POMDPs) have become a popular model for decision making under uncertainty in artificial intelligence (Kaelbling, Littman, & Cassandra 1998). A POMDP consists of: a set of states $S$; a set of actions $A$; a set of observations $Z$; a dynamics model that represents the probability of making a transition to state



[1] In some prior work(Brooks *et al.* 2006; Thrun 2000) a special exception is included to encode boundary conditions such as obstacles in a robotic task.

$s'$ after taking action $a$ in state $s$, $p(s'|s,a)$; an observation model describing the probability of receiving an observation $z$ in state $s$, $p(z|s)$; a reward model that specifies the reward received from being in state $s$ and taking action $a$, $R(s,a)$; the discount factor to trade off the value of immediate and future rewards, $\gamma$; and an initial belief state distribution, $b_o$.

A belief state $b_t$ is used to summarize the probability of the world being in each state given the past history of observations and actions $(o_{1:t}, a_{1:t})$. A policy $\pi : b \rightarrow a$ maps belief states to actions. The goal of POMDP planning techniques is to construct a policy that maximizes the (possibly discounted) expected sum of rewards $E[\sum_{t=1}^{T} \gamma^t R(s_t, a_t)]$ over an action sequence of length $T$. The policy is often found by computing this expected reward using a value function over the space of belief states. As the space of possible belief states is infinite, the value function cannot be represented by enumeration.

The POMDP formulation described above is agnostic about whether the underlying world states, actions, and observations are discrete or continuous. In the case where $S$, $A$, and $Z$ are discrete, Sondik (Sondik 1971) showed that the optimal finite horizon value function is piecewise linear and convex (PWLC) in the belief space and can therefore be represented by a finite set of $|S|$-dimensional $\alpha$-vectors. Each $\alpha$-vector corresponds to a "policy tree" specifying conditional sequences of actions, which depend on the observations received: $\alpha(s)$ is the value of executing this tree in state $s$. Therefore the expected value of starting at belief $b$ and following policy tree $j$ is computed by calculating the expected value of $\alpha_j$ under the distribution $b$, $\langle b, \alpha_j \rangle$ which is equal to $\sum_{s \in S} \alpha_j(s) b(s)$ for discrete $S$. Given an optimal value function represented by a set of $\alpha$-vectors, for a given belief $b$ the optimal action is chosen by selecting the action associated with the $\alpha$-vector that maximizes $\langle b, \alpha_j \rangle$.

In exact POMDP solutions, the number of $\alpha$ vectors required may grow exponentially with the length of the horizon, and this intractable growth often occurs in practice. Therefore the majority of prior work has focused on approximate solution techniques. Point-based value iteration (Pineau, Gordon, & Thrun 2006; Spaan & Vlassis 2005; Zhang & Zhang 2001) is one class of approximation techniques that exploits the piecewise linear and convex nature of the optimal discrete state value function. Point-based techniques estimate the value function at only a small set of $N$ chosen belief points $\tilde{B}$, resulting in a value function represented by at most $N$ $\alpha$-vectors. This representation is constructed by iteratively computing an approximately optimal $t$-step value function $V_t$ from the previously-computed $(t-1)$-step value function $V_{t-1}$ by backing up the value function at beliefs $b \in \tilde{B}$ using the Bellman equation:

$$V_t(b) = \max_{a \in A} \sum_{s \in S} R(s,a) b(s) + \ldots$$

$$\gamma \sum_{z \in Z} \max_{\alpha_{t-1} \in V_{t-1}} \sum_{s \in S} \sum_{s' \in S} p(s'|s,a) p(z|s') \alpha_{t-1}(s') b(s)$$

$$= \max_{a \in A} \left[ \sum_{s \in S} R(s,a) b(s) + \gamma \sum_{z \in Z} \max_{\alpha_{azj}} \langle \alpha_{azj}, b \rangle \right] \quad (1)$$

where $\alpha_{azj}$ is the vector associated with taking action $a$, receiving observation $z$ and then following the $(t-1)$ policy associated with $\alpha_j$. It is efficient to compute the dominant $\alpha$-vector at each $b$, and those vectors taken together provide an approximation of $V_t$ over entire belief space. Due to the PWLC property of discrete state POMDPs, this approximate representation is guaranteed to be a lower bound on the optimal value function. Different point-based techniques include carefully adding new elements to $\tilde{B}$ to improve this lower bound (Pineau, Gordon, & Thrun 2006) or updating a subset of $\tilde{B}$ at each iteration (Spaan & Vlassis 2005).

## 3 Switching State-Space Dynamics Models

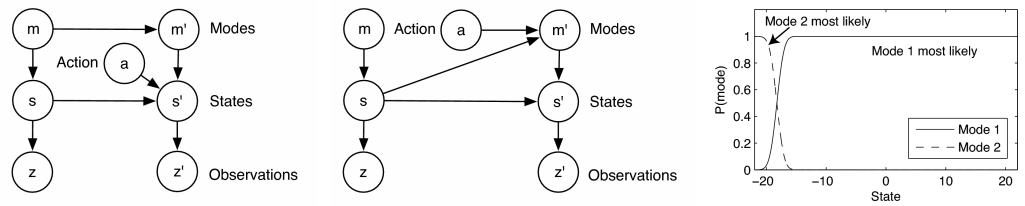
Our interest lies in using the rich framework of POMDPs to handle continuous-state problems directly without converting to a discrete representation. One critical representational issue is how to flexibly represent the dynamics in a continuous-state system. Previous approaches to planning in continuous-state POMDPs have frequently used a single linear Gaussian model to represent the new state distribution after an action. In general the dynamics of robotic grasping and many other problems of interest are highly complex and nonlinear. However we can approximate such dynamics using a "switching state space" model. This model will allow us to both represent actions that result in multimodal stochastic distributions over the state space, and succinctly represent any shared dynamics among states.

Switching state-space models (SSM) (also known as hybrid models and jump-linear systems) are a popular model in the control community for approximating systems with complex dynamics (Ghahramani & Hinton 2000). Typically an SSM consists of a set of linear state transition models. At each time step a hidden discrete mode state indexes a particular transition model which is used to update the hidden continuous-state vector. Frequently, the transition dynamics of the mode states are modeled as a Markov process (Ghahramani & Hinton 2000) (see figure 1(a) for an illustration). SSMs have been used to approximate the dynamics of a diverse set of complex systems, including planetary rover operation (Blackmore *et al.* 2007). The bulk of prior similar work on SSMs has focused on model parameter learning, and we are not aware of any prior work on using these models for POMDP planning tasks.

In order to model systems involving multi-modal, state-dependent dynamics (such as on sand vs concrete), we create a particular variant of an SSM that conditions the mode transitions on the previous continuous-state, similar to (Blackmore *et al.* 2007). Figure 1(b) displays a graphical model of the dynamics model used in this paper, which can be expressed as

$$p(s'|s,a) = \sum_{h} p(s'|s, m' = h) p(m' = h|s,a) \quad (2)$$

where $s, s'$ are the continuous states at time $t$ and $t+1$ respectively, $a$ is the discrete action taken at time $t$, and $m'$ is the discrete mode at time $t+1$. In this paper we will assume that for each action $a$ the hidden mode $m$ can take on one of $H$ values. Each mode value $h$ and action $a$ is associated with




(a) Typical SSM (b) Our Dynamics Model (c) Example of mode model

Figure 1: Switching State-Space Models

a linear Gaussian model $\mathcal{N}(s'; \zeta_{ha}s + \beta_{ha}, \sigma^2_{ha})$. For mathematical convenience we model the conditional probability of a mode taking on a particular value $h$ given the previous continuous-state $s$ and action $a$ using a weighted sum of $F$ Gaussians

$$p(m' = h|s, a) = \sum_{f=1}^{F} w_{fha}\mathcal{N}(s; \mu_{fha}, \sigma^2_{fha}). \quad (3)$$

This representation is slightly unusual since we are expressing the probability of a discrete variable $m$ conditioned on a continuous variable $s$. Note that for finite $H$ it is impossible to select the parameters $w_{fha}, \mu_{fha}, \sigma^2_{fha}$ such that the sum of probability of the next mode state $m'$ taking on any value for a given state $s$, $\sum_h p(m' = h|s, a)$, sums to 1 for all states $s$. Therefore in practice we will choose models that approximately sum to 1 over all the states of interest in a particular experimental domain. We choose to make this slightly awkward representational choice rather than normalizing the distribution across modes (such as by using a softmax function) because it allows closed form updates of the belief state and value function, as will be shown in the following sections. See figure 1(c) for an example mode model $p(m|s)$.

Substituting equation 3 into equation 2, the full dynamics model is a sum of Gaussian products:

$$p(s'|s, a) = \sum_{h=1}^{H} \mathcal{N}(s'; \zeta_{ha}s + \beta_{ha}, \sigma^2_{ha}) \sum_{f=1}^{F} w_{fha}\mathcal{N}(s; \mu_{fha}, \sigma^2_{fha}).$$

An added benefit of this model is that it can flexibly represent relative transitions (transitions that are an offset from the current state, by setting $\zeta \neq 0$) and absolute transitions (transitions that go to some arbitrary global state, by setting $\zeta = 0$ and $\beta \neq 0$). This allows the model to compactly represent domains in which many states share the same relative or absolute transition dynamics.

## 4 Point-Based POMDP Planning with Hybrid Models

We now describe a new planning algorithm for POMDPs with hybrid dynamics models. Recently Porta et al. (Porta *et al.* 2006) showed that for a continuous-state space $S$ and discrete actions $A$ and observations $Z$, the optimal finite horizon value function is piecewise linear and convex and may be represented by a finite set of $\alpha$-functions[2]. Therefore point-based approaches to continuous state POMDPs that exactly represent the $\alpha$-functions will also provide a lower bound on the value function. Porta et al.'s algorithm provides an approximation of a lower bound on the value function: our algorithm is inspired by theirs and handles multi-modal state-dependent dynamics and formulates the approximation step as a constrained optimization problem.

For clarity we will explain the mathematics for a one-dimensional state space, but it is easily extended to higher dimensions. We will assume that the reward model $r(s, a)$ and observation model $p(z|s)$ are represented by a weighted sum of Gaussians. To be precise, we assume the reward function $r(s, a)$ is expressed as a sum of $G$ Gaussian components for each action $a$, $r(s, a) = \sum_{g=1}^{G} w_{ag}\mathcal{N}(s; \mu_{ag}, \sigma^2_{ag})$, and each discrete observation $z \in Z$ is expressed as a sum of $L$ Gaussian components $p(z|s) = \sum_{l=1}^{L} w_{zl}\mathcal{N}(s; \mu_{zl}, \sigma^2_{zl})$ such that $\forall s \sum_z p(z|s) = 1$. Here we have assumed an observation model very similar to the mode representation (equation 3) and the same comments made for that choice apply here to the observation model. We choose to represent the belief states $b$ and $\alpha$-functions using weighted sums of Gaussians. Each belief state $b$ is a sum of $D$ Gaussians $b(s) = \sum_{d=1}^{D} w_d\mathcal{N}(s; \mu_d, \sigma^2_d)$, and each $\alpha_j$, the value function of policy tree $j$, is represented by a set of $K$ Gaussians $\alpha_j(s) = \sum_k w_k\mathcal{N}(s; \mu_k, \sigma^2_k)$. Recall that for each action $a$, there are $H$ modes and $F$ Gaussian components per mode.

We do not lose expressive power by choosing this representation because a weighted sum of a sufficient number of Gaussians can approximate any continuous function on a compact interval (and our domains of interest are closed and bounded and therefore fulfill the criteria of being compact). But, of course, we will be effectively limited in the number of components we can employ, and so in practice our models and solutions will both be approximations.

### 4.1 Belief Update and Value Function Back ups

Point-based POMDP planners must include a method for backing up the value function at a particular belief $b$ (as in equation 1) and for updating the belief state $b$ after a new action $a$ is taken and a new observation $z$ is received. Choosing a weighted sum of Gaussians representation allows both computations to be performed in closed form.

The belief state is updated using a Bayesian filter:

$$\begin{aligned} b^{a,z=i}(s) &= p(s'|z = i, a, b) \\ &\propto p(z = i|s', a, b)p(s'|a, b) \\ &= p(z = i|s')p(s'|a, b) \end{aligned}$$

[2]The expectation operator $\langle f, b\rangle$ is a linear function in the belief space and the value function can be expressed as the maximum of a set of these expectations: for details see (Porta *et al.* 2006).

where the last equality holds due to the Markov assumption. We compute the update by substituting in the dynamics and observation models, and normalizing such that $\int_s b^{a,z}(s)ds = 1$, yielding

$$b^{a,z=i}(s) = \sum_{dfhl} d_{dfhl} N(s|\mu_{dfhl}, \sigma^2_{dfhl}) \ .$$

Hence the representation of the belief as a mixture of Gaussians is closed under belief update.

The other key computation is to be able to back up the value function for a particular belief. Since we also use discrete actions and observations, this can be expressed as a slight modification to equation 1 by replacing sums with integrals and writing out the expectation:

$$\begin{aligned} V_t(b) &= \max_{a\in A} \int_{s\in S} R(s,a)b(s)ds + \gamma \sum_{z\in Z} \max_{\alpha_{azj}} \int_s \alpha_{azj} b(s) ds \\ &= \langle \max_{a\in A} R(s,a) + \gamma \sum_{z\in Z} \max_{\alpha_{azj}} \alpha_{azj}, b\rangle \end{aligned}$$

where we have used the inner-product operator $\langle f, b\rangle$ as shorthand for expectation to obtain the second equality. As stated previously, $\alpha_{azj}$ is the $\alpha$-function for the conditional policy corresponding to taking action $a$, receiving observation $z$ and then following the previous $t-1$-step policy tree $\alpha_j$, and can here be expressed as

$$\alpha_{azj}(s) = \int_{s'} \alpha_{j,t-1}(s') p(z|s') p(s'|s,a) ds'.$$

Substituting in all the chosen representations yields

$$\begin{aligned} \alpha_{azj}(s) &= \int_{s'} \sum_{k=1}^{K} w_k \mathcal{N}(s'; \mu_k, \sigma_k^2) \sum_{l=1}^{L} w_l \mathcal{N}(s'; \mu_l, \sigma_l^2) \dots \\ &\sum_{h=1}^{H} \mathcal{N}(s'; \zeta_{ha} s + \beta_{ha}, \sigma^2_{ha}) \sum_{f=1}^{F} w_{fha} \mathcal{N}(s; \mu_{fha}, \sigma^2_{fha}) ds' \\ &= \sum_{f=1}^{F}\sum_{h=1}^{H}\sum_{k=1}^{K}\sum_{l=1}^{L} w_{fha} w_k w_l \mathcal{N}(s; \mu_{fha}, \sigma^2_{fha}) \dots \\ &\int_{s'} \mathcal{N}(s'; \mu_k, \sigma_k^2) \mathcal{N}(s'; \mu_l, \sigma_l^2) \mathcal{N}(s'; \zeta_{ha} s + \beta_{ha}, \sigma^2_{ha}) ds'. \end{aligned}$$

To perform the integral we combine the three Gaussians inside the integrand into a single Gaussian which is a function of $s'$ and other terms that are independent of $s'$. Integrating over $s'$ yields

$$\alpha_{azj} = \sum_{f=1}^{F}\sum_{h=1}^{H}\sum_{k=1}^{K}\sum_{l=1}^{L} w_{fhkl} \mathcal{N}(s; \mu_{fhkl}, \sigma^2_{fhkl})$$

where

$$\begin{aligned} w_{fhkl} &= w_{fha} w_k w_l \mathcal{N}(s_l; s_k, \sigma_k^2 + \sigma_l^2) \dots \\ &\mathcal{N}(\mu_{fha}; \frac{c - \beta_{ha}}{\gamma_{ha}}, \sigma^2_{fha} + \frac{C + \sigma^2_{ha}}{\gamma^2_{ha}}), \end{aligned}$$

$$C = \frac{\sigma_l^2 \sigma_k^2}{\sigma_l^2 \sigma_k^2},\ \mu_{fhkl} = \frac{(C+\sigma^2_{ha})\mu_{fha} + \sigma^2_{fha}\gamma_{ha}(c - \beta_{ha})}{\sigma^2_{fha}(C + \sigma^2_{ha})},$$

$$\sigma^2_{fhkl} = \frac{\sigma^2_{fha}(C + \sigma^2_{ha})}{C + \sigma^2_h + \sigma^2_{fha}\gamma^2_{ha}},\ c = \frac{\mu_k \sigma_l^2 + \sigma_k^2 \mu_l}{\sigma_l^2 \sigma_k^2}.$$

The new $\alpha_{azj}$ now has $F \times H \times K \times L$ components, compared to the step $t-1$ $\alpha_j$, policy tree, which only had $K$ components. To finish the value function backup, we substitute $\alpha_{azj}$ back into equation 4 and choose the policy tree $\alpha$ that maximizes the future expected reward $\langle \alpha, b\rangle$ of belief $b$

$$\begin{aligned} \alpha(s) &= \max_a R(s,a) + \gamma \sum_{z=1}^{|Z|} \max_{\alpha_{azj}} \alpha_{azj} \\ &= \max_a \left[ \sum_{g=1}^{G} N(s|\mu_g, \sigma_g^2) + \gamma \sum_{z=1}^{|Z|} \max_{\alpha_{azj}} \alpha_{azj} \right] \end{aligned}$$

Since all elements in this result are weighted sums of Gaussians, the $\alpha$ function stays in closed form. Note that had we utilized a softmax distribution for the observation model or mode probabilities that it would not be possible to iteratively perform the integrals in closed form.

Unfortunately, the number of Gaussian components in a single $\alpha$ function has greatly increased: from $K$ to $G + |Z|FHKL$ components. Compared to previous approaches (Porta *et al.* 2006) the new dynamics model has introduced an extra factor of $FH$ components. The number of components therefore scales exponentially in the time horizon with base $|Z|FHL$.

### 4.2 Approximating the $\alpha$ functions

It is not computationally feasible to maintain all components over multiple backups. Instead, by carefully combining the components generated after each backup, we maintain a bounded set of $\alpha$ functions. Since $\alpha$ functions represent the value of executing a particular policy tree over the entire belief space, it is important to make the approximation as close as possible throughout the belief space.[3] To reduce the number of components used to represent the belief states and $\alpha$ functions, Porta et al. use a slight variant of Goldberger and Roweis's method (Goldberger & Roweis 2005) that minimizes the Kullback-Leibler (KL) distance between the original model $f(x)$ and the approximation $\tilde{f}(x)$

$$D_{KL}(f||\tilde{f}) = \int_x f(x) \log \frac{f(x)}{\tilde{f}(x)} dx. \qquad (4)$$

However, the KL distance is not particularly appropriate as a distance measure for quantities that are not probability distributions. It also can result in poor approximations in parts of the space where the original function has small values since if $f(x)$ is zero then regardless of the value of $\tilde{f}(x)$ the distance for that $x$ is always zero. An alternate distance measure without these shortcomings is the L2 norm: a small value means a good approximation $\tilde{\alpha}$ of the value function over the entire belief space.

The L2 norm, or sum squared error, between two weighted sums of Gaussians is:

$$= \int_s \left[ \sum_i^M w_i \mathcal{N}(s; \mu_i, \sigma_i^2) - \sum_j^N w_j \mathcal{N}(s; \mu_j, \sigma_j^2) \right]^2 ds$$

[3] Ideally we could restrict this approximation to the reachable belief space; however analyzing the reachable belief space in continuous-state POMDPs will be an area of future work.

$$= \sum_i^M \sum_{i'}^M w_i w_{i'}' \mathcal{N}(\mu_{i'}; \mu_i, \sigma_i^2 + \sigma_{i'}^2) + \sum_j^N \sum_j'^N w_j w_{j'}' \mathcal{N}(\mu_{j'}; \mu_j, \sigma_j^2 + \sigma_{j'}^2) - 2 \sum_j^N \sum_i^M w_j w_i \mathcal{N}(\mu_i; \mu_j, \sigma_j^2 + \sigma_i^2).$$

There is no analytic solution for the parameters $w_i, \mu_i, \sigma_i^2$ that minimizes this expression, and numerical approaches such as gradient descent are challenged by the high dimensionality of the parameter space. However we can approximate this optimization using Zhang and Kwok's recent work on reducing the number components in kernel function mixture models (Zhang & Kwok 2006). This work minimizes an upper bound on the L2 norm by clustering the original components into small groups, and fitting a single weighted Gaussian to each group. More precisely, the L2 norm can be upper bounded by a function of the L2 error of each cluster:

$$L2 \leq M \sum_{i=1}^{M} \int \left[ w_i \mathcal{N}(s; \mu_i, \sigma_i^2) - \sum_{j \in S_i} w_j \mathcal{N}(s; \mu_j, \sigma_j^2) \right]^2 ds$$

where $S_i$ is the $i$-th cluster and $M$ is the total number of components in the approximation $\tilde{\alpha}$. The parameter fitting procedure is simplified from the general problem of gradient descent by the clustering procedure. The complete procedure can be performed iteratively, by creating clusters through assigning all components in the original function to their closest (in the L2 norm sense) component in the approximation, refitting a single component for each cluster, and repeating.

Although the upper bound to the L2 norm can be easily optimized, we would also like to constrain the exact L2 norm error to be within some threshold. Therefore we formulate our approximation step as a constrained optimization problem, using the objective function from Zhang and Kwok, but requiring that the final approximation $\tilde{\alpha}$ both lies below a maximal L2 norm threshold, and contains no more than a fixed maximum number of components, $M_{max}$. This can be stated mathematically as follows:

$$\arg \min_{w_i, \mu_i, \sigma_i^2, M} M \sum_{i=1}^{M} \int \left[ w_i \mathcal{N}(s; \mu_i, \sigma_i^2) - \sum_{j \in S_i} w_j \mathcal{N}(s; \mu_j, \sigma_j^2) \right]^2 ds$$
$$s.t. ||\tilde{\alpha} - \alpha||_2 \leq t \,\&\, M \leq M_{max}$$

where $t$ is L2 norm threshold.

We initialize the components of the approximation with a random subset of the original components. The remaining original components are then clustered to their closest (in the L2 norm) component in the approximation, and then a single component is refit for each cluster. In order to ensure the parameter initialization lies within the feasible region spanned by the constraints, we compute the L2 norm of this initialization, and discard solutions that do not satisfy the L2 norm constraint. Note that this also provides a principled mechanism for selecting the number of components constituting the approximation: the number of components $M$ in the approximation is increased until either an initialization is found that lies within the feasible region of the constraints, or $M$ no longer satisfies the second constraint. If both constraints cannot be satisfied, $M$ is set to $M_{max}$.

Once the parameters are initialized and $M$ is fixed, we follow Zhang and Kwok's procedure for optimizing.

For some problems it is computationally demanding to even evaluate whether the L2 norm constraint has been violated. When the original number of components $N$ is sufficiently large, computing the component of the L2 norm arising from the product of the original $\alpha$-function with itself (an O($N^2$) computation) is very expensive. In this situation we change the constraints of the problem to

$$\langle \tilde{\alpha}, b \rangle - \langle \alpha_{old}, b \rangle > 0 \,\&\, M \leq M_{max}.$$

Recall that the value function $\alpha$ is being backed up for a particular belief state $b$. A minimum requirement for the approximation $\tilde{\alpha}$ is that the value of this belief under $\tilde{\alpha}$ must be greater than or equal to its value prior to the backup $\langle \alpha_{old}, b \rangle$, otherwise the approximation $\tilde{\alpha}$ has failed to capture a new better policy for this belief state, and will not be added to the pool of $\alpha$-functions that represent the value function. This quantity is fast to compute and serves as an alternate constraint when computing the full L2 norm is computationally intractable due to the size of N.

### 4.3 Planning

We now have the major components necessary to apply a point-based approach to POMDP planning. First a set of belief points $B$ is selected by executing random actions from an initial starting belief state $b_0$. We initialize the value function as a single $\alpha$ function (see details below). Starting with this initial $\alpha$ function, we iteratively perform value function backups for the chosen belief set $\tilde{B}$. Our point-based approach closely matches Perseus (Spaan & Vlassis 2005) which does not update all belief points at each round, and maintains a fixed belief point set. Perseus operates by selecting belief points randomly from $\tilde{B}$ and backing them up until the newly constructed $\alpha_t$ functions have improved or maintained the previous $V_{t-1}$ value for all $b \in \tilde{B}$. Then all previous $\alpha_{t-1}$ functions are discarded and only the new $\alpha_t$ functions created are used for the next backups round.

## 5 Experiments

To demonstrate the benefit of this approach we ran our algorithm on two illustrative one-dimensional problems. Both domains exhibit state-dependent dynamics single linear model. In the first example we motivate the need for continuous-state representations by providing an example where our continuous-state approach outperforms a discrete-state POMDP solver on a task where the fine granularity needed to perform well creates a very large discrete-state space. In the second example we demonstrate the need for hybrid models in order to handle tasks with multi-modal dynamics. In both examples the maximum number of components allowed per $\alpha$-function $M_{max}$ was 200 and the L2 norm threshold $t$ was set at 10.

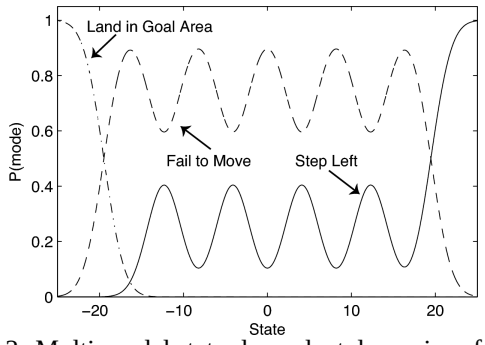


Figure 2: Multi-modal state-dependent dynamics of $Jump$. A given mode's probability (such as step left) varies over the state space, and more than one mode is possible for a state.

### 5.1 Power Supply Hunting: Variable Resolution Navigation

We first demonstrate that when fine resolution is required to execute a good policy, the continuous-state approach can outperform the discrete-state POMDP solver Perseus on a task with state-dependent dynamics. Here a robot must navigate a long corridor ($s \in$ [-21,21]) to find a power socket which is located at -16.2. The robot can move left or right using small or large steps that transition it 0.1 or 5.0 over from its current location plus Gaussian noise of standard deviation 0.01. If the robot tries to move too close to the left or right wall it will bump into the wall. The robot can also try to plug itself in, which leaves the robot at the same location. All movement actions receive a reward of 0.05. If the robot plugs itself in at the right location it receives a large positive reward (modeled by a highly peaked Gaussian); otherwise it receives a lesser reward of 5.8. The power supply lies beneath the robot sensors so the robot is effectively blind.

We compared three planning approaches for this task: a discrete-state planner, a linear-model continuous-state planner, and our hybrid-model continuous-state planner. Our algorithm and the linear-model continuous-state planner were trained using 1000 belief points gathered by starting in a random state $s \in [-21, 21]$ with a Gaussian approximately uniform belief and acting randomly for episodes of 30 steps. The robot can always achieve at least the reward associated with only executing $PlugIn$. Therefore we used the $PlugIn$ action reward function, scaled to account for infinite actions and discounting (multiplied by $1/(1-\gamma)$) as the initial lower bound value function.

We created 4 resolutions of uniformly spaced grids of the domain and used the discrete- state value iteration technique Perseus to solve each discrete version of the task.

The value functions produced by each planner were tested by computing the average reward received over 50 episodes of 100 steps/episode using the policy associated with the $\alpha$-functions/vectors in the value function. At the start of each episode the robot is placed at a state $s$ that is chosen randomly from the uniform distribution spanning [-19,19]. The robot's belief state is initialized to be a set of 4 high variance Gaussians spanning the state space. See Table 1 for results.

Our hybrid model finds a good policy that involves taking big and small actions to first localize the belief (at the wall) and then taking three more steps to reach the power socket. The linear model continuous-state POMDP runs faster but fails to find a good policy since it cannot model the state-dependent dynamics near the wall edges, and there are no unique observations. The discrete-state solutions does poorly at coarse granularities since the $PlugIn$ reward gets washed out by averaging over the width of a too wide state. At fine state granularities the discrete approach finds a good policy but require more time: our continuous-state planner finds a solution faster than the coarsest discrete-state planner that can find a good solution. It is also important to note that it is hard to determine a priori what level of discretization is required for a discrete-state planner to find a good policy, and choosing a conservatively fine discretization can result in a substantially longer planning time.

| Models | Continuous-state | | Discretized (Number of States) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Linear | Hybrid | 840 | 1050 | 1155 | 1208 | 1260 | 1470 | 2100 |
| Time(s) | 112 | **2473** | 665 | 1261 | 1685 | **2751** | 2939 | 4155 | 19438 |
| Reward | 290 | 465* | 290 | 290 | 290 | 510* | 488* | 488* | 488* |

Table 1: Power Supply Experiment Results. *No significant difference as measured by 3 t-tests ($p > 0.05$) between the rewards received from running the hybrid-planner policy and the 3 discrete-state policies respectively.

### 5.2 Locomotion over Rough Terrain: Actions with Bimodal Next State Distributions

Our second example presents a domain where the dynamics are state-dependent and multi-modal, and therefore poorly represented by a unimodal linear model. In robotic legged locomotion over rough terrain (such as DARPA's LittleDog project) a robot may need to traverse an uneven rocky surface to reach a goal location. Our example is inspired by this problem. The robot starts on an initial flat surface and must traverse an area with 4 rocks separated by sand to reach a goal location. At each step the robot can attempt to step forward or signal it is done. The robot is faulty and works best on hard surfaces: at each step it may succeed in executing a step or stay in the same place. Figure 2 displays the multi-modal state-dependent nature of the dynamics. The robot is receives a relatively very low reward for stepping into the sand, and a medium reward for each step on the rocks. A $Signal$ action results in a large reward if the robot has reached the final location, and a small reward otherwise. The observation model provides a noisy estimate of where the robot is (sand, rock 1-4, start or finish).

We tested our hybrid model and a linear model that averages the bimodal distributions. The models were tested in a manner similar to the prior experiment (except using 100 beliefs). The agent can always perform at least as well as performing the $Signal$ action forever so the $Signal$ action reward was used as an initial value function, scaled to account for discounting and performing the action indefinitely. The following table displays the results:

| | Average Total Reward Received |
|---|---|
| Hybrid Model | 8055.5 |
| Linear Model | 3452.3 |

Since the hybrid model can correctly represent that a step from the initial platform will keep the robot on the platform or move it to the first rock, it can find a good policy of repeatedly trying to step and will eventually reach the goal platform. In contrast, the linear model performs poorly because its dynamics model leads it to believe that stepping from the platform will result in landing in a relatively undesirable and hard to escape sandpit. Instead the linear model policy simply signals immediately.

## 6 Discussion and Future Work

Our preliminary experiments demonstrate that our approach produces good plans in tasks not amenable to linear-model continuous-state approaches. The approach also outperformed a discrete-state planning approach in a task requiring a fine grained resolution. Our approach is unlikely to perform well in domains with many extremely sharp transitions in its dynamics model since this is expensive to model using a weighted sum of Gaussians, and may be represented using a discrete model.

In the future we intend to apply this approach to higher dimensional problems, such as robotic legged locomotion. We would also like to further investigate and refine the parameter settings and approach used to perform the approximation step, as well as to compare to more recent discrete-state POMDP planners such as FSVI(Shani, Brafman, & Shimony 2007). However, this representation has been sufficient to demonstrate the potential value of using hybrid models in a continuous-state POMDP framework for solving problems with nonlinear dynamics.

## References


Blackmore, L.; Gil, S.; Chung, S.; and Williams, B. 2007. Model learning for switching linear systems with autonomous mode transitions. In *IEEE Conference on Decision and Control*.

Brooks, A.; Makarenko, A.; Williams, S.; and Durrant-Whyte, H. 2006. Parametric POMDPs for planning in continuous state spaces. *Robotics and Autonomous Systems*.

Ghahramani, Z., and Hinton, G. 2000. Variational learning for switching state-space models. *Neural Computation* 12:831–864.

Goldberger, J., and Roweis, S. 2005. Hierarchical clustering of a mixture model. In *NIPS*.

Kaelbling, L.; Littman, M.; and Cassandra, A. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101:99–134.

Pineau, J.; Gordon, G.; and Thrun, S. 2006. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research* 27:335–380.

Porta, J.; Spaan, M.; Vlassis, N.; and Poupart, P. 2006. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research* 7:2329–2367.

Shani, G.; Brafman, R.; and Shimony, S. 2007. Forward search value iteration for POMDPs. In *IJCAI*.

Smith, T., and Simmons, R. 2004. Heuristic search value iteration for POMDPs. In *UAI*.

Sondik, E. J. 1971. *The Optimal Control of Partially Observable Markov Processes*. Ph.D. Dissertation, Stanford University.

Spaan, M., and Vlassis, N. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research* 24:195–220.

Thrun, S. 2000. Monte carlo POMDPs. In *NIPS*.

Zhang, K., and Kwok, J. 2006. Simplifying mixture models through function approximation. In *NIPS*.

Zhang, N., and Zhang, W. 2001. Speeding up the convergence of value iteration in partially observable markov decision processes. *Journal of Artificial Intelligence Research* 14:29–51.