Ann Math Artif Intell
DOI 10.1007/s10472-010-9202-1

Planning in partially-observable switching-mode continuous domains

Emma Brunskill · Leslie Pack Kaelbling · Tomás Lozano-Pérez · Nicholas Roy

© The Author(s) 2010. This article is published with open access at Springerlink.com

Abstract Continuous-state POMDPs provide a natural representation for a variety of tasks, including many in robotics. However, most existing parametric 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 dynamics model that can represent multi-modal state-dependent dynamics. We present the Switching Mode POMDP (SM-POMDP) planning algorithm for solving continuous-state POMDPs using this dynamics model. We also consider several procedures to approximate the value function as a mixture of a bounded number of Gaussians. Unlike the majority of prior work on approximate continuous-state POMDP planners, we provide a formal analysis of our SM-POMDP algorithm, providing bounds, where possible, on the quality of the resulting solution. We also analyze the computational complexity of SM-POMDP. Empirical results on an unmanned aerial vehicle collisions avoidance simulation, and a robot navigation simulation where the robot

This research was conducted while E. Brunskill was at the Massachusetts Institute of Technology.

E. Brunskill (✉)
Department of Electrical Engineering and Computer Science,
University of California, Berkeley, Berkeley, CA, USA
e-mail: emma@cs.berkeley.edu

L. P. Kaelbling · T. Lozano-Pérez · N. Roy
Computer Science and AI Laboratory (CSAIL),
Massachusetts Institute of Technology, Cambridge, MA, USA

L. P. Kaelbling
e-mail: lpk@csail.mit.edu

T. Lozano-Pérez
e-mail: tlp@csail.mit.edu

N. Roy
e-mail: nickroy@csail.mit.edu

Published online: 09 July 2010

has faulty actuators, demonstrate the benefit of SM-POMDP over a prior parametric approach.

Keywords Partially observable Markov decision process · Planning · Continuous-states

Mathematics Subject Classifications (2010) 90C40 · 68Txx · 68T37

1 Introduction

Partially observable Markov decision processes (POMDPs) [9] provide a rich framework for describing a number of planning problems that arise in situations with hidden state and stochastic actions. In many applications, such as autonomous navigation across varying terrain or robotic grasping, the world is most naturally represented using continuous states. Planning in continuous environments is challenging because a continuous domain with even a single dimension offers an infinite number of different world states. For example, an autonomous car driving on a straight road could take on an infinite range of values, and the planner must determine an action for each value. If the world state is unknown, the decision-making algorithm must consider a distribution over an infinite number of possible states, known as the *belief*, and then compute the expected benefit of different actions given different beliefs, known as the *value function*. In this work we are interested in planning algorithms for operating in such partially-observable, continuous-valued domains with complex dynamics.

Outside of the famous linear quadratic Gaussian controller (see Burl [3]), there has been relatively little work on continuous-state planning in partially observable environments. Brooks et al. [2] restrict the belief state to be a unimodal Gaussian,[1] and then plan by discretizing the Gaussian parameters and using Fitted Value Iteration over that representation. However, this approach will be limited in any domain that requires multi-modal beliefs to achieve good performance. In addition, the resulting plan quality depends on the resolution of the discretization of the parameter space, which is hard to know in advance and has a direct impact on computational cost. Thrun's Monte Carlo POMDP [21] approach can handle arbitrary dynamics models but requires often-expensive sampling to perform the belief updates and to compute the value function estimate. Recent work by Zhou et al. [24] generalizes the work of Brooks et al. to a more generic class of parametric families. While the authors present nice theoretical bounds on their approach, the cost scales as a function of the discretized parameter space, which may be expensive in domains that require a large number of parameters, or a fine discretization of the parameter space in order to achieve good empirical results.

This prior work demonstrates the interest in using parametric functions to encode the world models in continuous-domains; such functions provide a compact mechanism to represent distributions and values over infinite sets. Indeed, Porta et al. [15]

[1] A special exception is included to encode boundary conditions such as obstacles in a robotic task.

have demonstrated that using linear Gaussian models allows the main operators of POMDP planning, value function backups and belief updating, to be done in closed form. By optimizing and adjusting both the parameters and the value function, Porta et al.'s planner can automatically adapt the value function representation at each backup, introducing new components to the representation as the structure of the value function changes. In this sense it can be considered a variable resolution approach to planning (see for example Munos and Moore [11]), but one that dynamically determines the resolution as part of the standard value function computation. This can be a significant advantage over standard uniform resolution discrete-state POMDP planners, as Porta et al. demonstrated by showing their approach was faster than the discrete-state POMDP planner Perseus [20] in a simulation experiment.

However, if parametric functions are used to represent the world models and value function, it is also critical to ensure that the chosen functions are sufficiently expressive to adequately represent the problems of interest. The work by Brooks [2], Porta [15] and their colleagues presented results on problems that used the simple linear Gaussian to describe the effects of an action. However, the dynamics of robotic grasping, autonomous navigation over varying terrain, and many other problems of interest are highly complex and nonlinear. For example, an autonomous car crossing varied terrain will exhibit different dynamics depending on whether the ground underneath is sand or rocks. Though such dynamics are easily represented in discrete-state environments using 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 the Switching Modes POMDP (SM-POMDP) algorithm which uses a switching state space representation of the world dynamics to perform approximate closed form planning in continuous-valued environments. SM-POMDP extends Porta et al.'s [15] prior work on continuous-state POMDPs by using a model that can both represent actions that result in multimodal stochastic distributions over the state space, and succinctly represent any shared dynamics among states. SM-POMDP is suitable for a much larger class of POMDP problems than past related approaches, but still maintains the strengths of parametric representations in continuous-state POMDP planning.

In contrast to most[2] prior approximate continuous-state POMDP work [2, 15, 21], we also provide a theoretical analysis of several variants of our SM-POMDP planner, bounding where possible the error between the resulting SM-POMDP value function, and the optimal value function. We also analyze the computational complexity of planning using SM-POMDP. As is often the case, we show there is a tradeoff between computational tractability and guarantees on the resulting optimality of the produced plan. Our analysis could aid users in selecting the right variant of SM-POMDP appropriate for the tractability and optimality constraints of their own application.

Our paper proceeds as follows. In Section 2 we briefly provide an introduction to POMDPs, and then discuss switching state dynamics models in Section 3. We next present how to plan with these models in Section 4. To keep planning tractable, this section also presents several methods for ensuring that the value function

[2] A notable exception is the work of Zhou et al. [24] who provide an elegant formal analysis of their approach.

representation uses a bounded number of parameters. In Section 5 we analyze SM-POMDP in terms of its computational complexity, and, where possible, provide bounds on the quality of its solutions. Experimental results are given in Section 6 that demonstrate the advantage of SM-POMDP over prior parametric approaches on an unmanned aerial vehicle collisions avoidance simulation, and a robot navigation simulation where the robot has faulty actuators. Section 7 concludes.

2 POMDPs

Partially observable Markov decision processes (POMDPs) have become a popular model for decision making under uncertainty in artificial intelligence [9]. 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 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, γ; and an initial belief distribution, b_o.

A belief state $b_t = p(s|a_{1:t}, z_{1:t})$ is used to summarize the probability of the world being in each state given the past history up to the current time t of observations and actions $(z_{1:t}, a_{1:t})$. A policy $\pi : b \rightarrow a$ maps belief states to actions. The goal of POMDP planning 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 a value function over the space of beliefs that gives the expected sum of rewards for each state. As the space of possible beliefs is infinite, the value function cannot be represented by a table mapping states to values as is often done for Markov decision processes.

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 [19] 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 α-vectors. Each α-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 α_j under the distribution b, denoted by $\langle b, \alpha_j \rangle$, 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 α-vectors, for a given belief b the optimal action is chosen by selecting the action associated with the α-vector that maximizes the expected value. One common way to compute the α-vectors is to perform value iteration, which involves using a set of t-step conditional policy trees, each with an associated α-vector, to create new $(t + 1)$-step policy trees and α-vectors by considering all possible actions, and all possible observations, and then all possible next t-step policy trees that could follow a particular action, observation tuple. In infinite-horizon settings, the base α-vector can represent a lower bound value on a particular policy, such as executing the same action always. In such scenarios, the α-vector after T rounds of value iteration can be interpreted as following a particular

conditional policy tree for T steps, followed by the base policy of executing the same action from then onwards.

The number of different conditional policy trees, and hence the number of α-vectors required to express a POMDP value function exactly, can 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 [14, 20, 23] 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 α-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:

$$\begin{aligned} V_t(b) &= \max_{a \in A} \sum_{s \in S} R(s,a) b(s) + \gamma \sum_{z \in Z} \max_{\alpha_{t-1} \in V_{t-1}} \sum_{s \in S} \sum_{s' \in S} p(s'|s,a)\, p(z|s') \alpha_{t-1}(s') b(s) \\ &= \max_{a \in A} \left[\sum_{s \in S} R(s,a) b(s) + \gamma \sum_{z \in Z} \max_{\alpha_{azj}} \langle \alpha_{azj}, b \rangle \right] \end{aligned} \tag{1}$$

where α_{azj} is the vector associated with taking action a, receiving observation z and then following the $(t-1)$-step policy associated with α_j. We have used the notation $\langle x, y \rangle$ to denote the inner product between vectors x and y. It is efficient to compute the dominant α-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 [14] or updating a subset of $\tilde{B}$ at each iteration [20].

3 Switching state–space dynamics models

Our interest lies in using the rich framework of POMDPs to handle continuous-state problems with complicated dynamics directly without converting to a discrete representation.

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 [6]. Typically the state space of a SSM is the product of some continuous state variables, and a discrete mode variable. The corresponding dynamics are given by a set of linear state transition models. At each time step the discrete mode variable indexes a particular transition model which is used to update the continuous-state variables. Frequently, the transition dynamics of the mode states are modeled as a Markov process [6] (see Fig. 1a for an illustration). SSMs have been used to approximate the dynamics of a diverse set of complex systems,

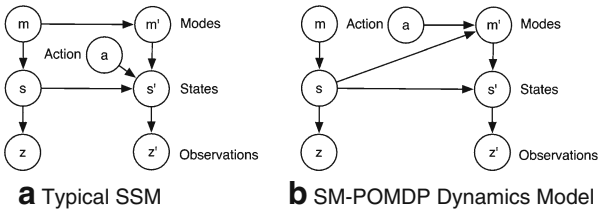


Fig. 1 Switching state–space models

including planetary rover operation [1], honeybee dances [12] and the IBOVESPA stock index [5]. The bulk of prior work on SSMs has focused on inference in SSMs with hidden state or mode variables, or model parameter learning. We are not aware of any prior work on using these models for POMDP planning tasks.

Despite this lack of prior work, there exist a number of planning domains where the environment can be well modeled by a SSM. For example, consider a quadruped robot that must navigate over rocks to cross a stream during a search and rescue mission. This task requires computing a sequence of actions which could include jumping or walking. However, the result of each action will depend on the current surface under the robot: jumping from any rock is likely to have the same relative effect, but trying to jump from within the stream is likely to fail. Here the dynamics of the world can be characterized as a switching state model, where the dynamics switch depending on the type of surface the robot is currently standing on. Depending on the sensors a robot is equipped with, the robot may not be able to perfectly resolve which surface it is presently on, and may instead need to select actions conditioned on a probability distribution over the two possible surfaces.

Another example from robotics is a sensorless robot attempting to navigate to find a power outlet along a long hallway. As the robot attempts to navigate to the left or right, its movements will typically succeed, unless it is at one of the two hallway ends, in which case the robot will stay in the same place. Here the robots dynamics depend on its (unknown) location along the hallway. The robot can use this knowledge to localize itself, by effectively trying to go consistently left until it is highly confident it has reached the left hallway end, at which point it can then navigate to the power outlet. These two examples are representative of the types of planning problems we seek to address in this paper, and we will describe some simulation results with these domains later on in the experimental section.

In order to model such systems which involve multi-modal, state-dependent dynamics we create a particular variant of an SSM that conditions the mode transitions on the previous continuous-state, similar to [1]. Figure 1b displays a graphical model of the dynamics model used in this paper, which can be expressed as

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

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 transition model for the continuous state variable, specifically 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

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

This representation is slightly unusual; we are expressing the probability of a discrete variable m conditioned on a continuous variable s. See Fig. 2 for an example mode model $p(m|s)$. Note that for finite F it is impossible to select the parameters $w_{fha}, \mu_{fha}, \sigma_{fha}^2$ such that the sum of probabilities of the next mode state m' taking on any value for a given state s, $\sum_h p(m' = h|s, a)$, equals 1 for all states s. Therefore in practice we will choose models that approximately sum to one over all the states of interest in a particular experimental domain. We choose to make this alternate representational choice rather than using a softmax function over the state space because the mixture of Gaussians representation will allow closed form updates of the belief state and value function, as will be shown in the following sections. However, this choice comes with two notable shortcomings: it is typically non-trivial to specify the parameters to ensure the model approximately sums to one over the state space of interest, and the number of components required can be large.

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

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

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 $\beta \neq 0$. The model can therefore compactly represent state dynamics in which many states share the same relative or absolute transition dynamics. Discrete approaches must enumerate over all discrete states the probability of each state transitioning to

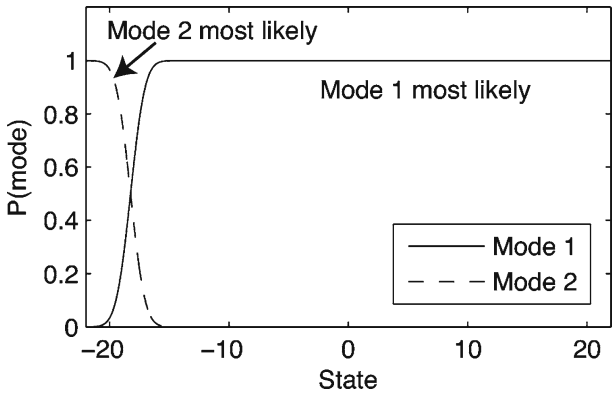


Fig. 2 Example of a mode model $p(m|s)$

each other state, even if many states share the same dynamics (such as "move 2 grid cells to the left"). In contrast, our model can represent such dynamics by a single linear Gaussian $\mathcal{N}(s'; s-2, \sigma^2)$.

4 SM-POMDP

We now describe a new planning algorithm for POMDPs with switching-mode dynamics models. Recently Porta et al. [15] 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 α-functions.[3] Therefore point-based approaches to continuous state POMDPs that represent α-functions at a finite set of beliefs 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.

For clarity we will explain the algorithm 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, $r(s,a) = \sum_{g=1}^{G} w_{ag}\mathcal{N}(s; \mu_{ag}, \sigma_{ag}^2)$, and each discrete observation $z \in Z$ is expressed as a sum of L Gaussian components $p(z|s) = \sum_{l=1}^{L} w_{zl}\mathcal{N}(s; \mu_{zl}, \sigma_{zl}^2)$ such that $\forall s \sum_z p(z|s) \approx 1$. Here we have assumed an observation model very similar to the mode representation (3) and the same comments made for that choice apply here to the observation model.

We also choose to represent the belief states b and α-functions using weighted sums of Gaussians. Each belief state b is a sum of D Gaussians $b(s) = \sum_{d=1}^{D} w_d\mathcal{N}(s; \mu_d, \sigma_d^2)$, and each α_j, the value function of policy tree j, is represented by a set of K Gaussians $\alpha_j(s) = \sum_k w_k\mathcal{N}(s; \mu_k, \sigma_k^2)$. Recall that for each action a, there are H modes and F Gaussian components per mode. Figure 3 lists the models and symbols used to describe a SM-POMDP.

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) (see for example Park and Sandberg [13]). 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 updates and values function backups

Point-based POMDP planners must include a method for backing up the value function at a particular belief b (as in (1)) and for updating the belief state b after a new action a is taken and a new observation z is received. Representing all parameters

[3] 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 [15].

- $b(s) = \sum_{d=1}^{D} w_d \mathcal{N}(s; \mu_d, \sigma_d^2)$ is the belief state
- m is the discrete hidden mode and there are H modes for each action a
- $p(s'|s, a) = \sum_{h=1}^{H} p(s'|s, m' = h) p(m' = h|s, a)$ is the transition model for action a where:
  - $p(s'|s, m' = h) = \mathcal{N}(s'; \zeta_{ha} s + \beta_{ha}, \sigma_{ha}^2)$ is the linear Gaussian model for the discrete mode h for action a, and
  - $p(m' = h|s, a) = \sum_{f=1}^{F} w_{fha} \mathcal{N}(s; \mu_{fha}, \sigma_{fha}^2)$ represents the probability of mode h given state s and action a
- $p(z|s) = \sum_{l=1}^{L} w_{zl} \mathcal{N}(s; \mu_{zl}, \sigma_{zl}^2)$ is the observation model for observation z
- $r(s, a) = \sum_{g=1}^{G} w_{ag} \mathcal{N}(s; \mu_{ag}, \sigma_{ag}^2)$ is the reward model for action a
- $\alpha(s) = \sum_{k=1}^{K} w_k \mathcal{N}(s; \mu_k, \sigma_k^2)$ is an α-function

Fig. 3 SM-POMDP models and symbols

of the value function as a weighted sum of Gaussians allows both computations to be performed in closed form.

The belief state is updated using a Bayesian filter:

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

where the last equality holds due to the Markov assumption. We compute the update by substituting in the dynamics and observation models:

$$\begin{aligned} b^{a,z=i}(s) \propto \sum_{l=1}^{L} w_l \mathcal{N}\left(s'; \mu_l, \sigma_l^2\right) \int_s \sum_{h=1}^{H} \mathcal{N}\left(s'; \zeta_{ha} s + \beta_{ha}, \sigma_{ha}^2\right) \\ \times \sum_{f=1}^{F} w_{fha} \mathcal{N}\left(s; \mu_{fha}, \sigma_{fha}^2\right) \sum_{d=1}^{D} w_d \mathcal{N}\left(s; \mu_d, \sigma_d^2\right) ds. \end{aligned}$$

After pulling the sums outside the integral, it is necessary to integrate the product of three Gaussians within the integral:

$$\begin{aligned} b^{a,z=i}(s) \propto \sum_{d,f,h,l} w_d w_{fha} w_d \mathcal{N}\left(s'; \mu_l, \sigma_l^2\right) \int_s \mathcal{N}\left(s'; \zeta_{ha} s + \beta_{ha}, \sigma_{ha}^2\right) \\ \times \mathcal{N}\left(s; \mu_{fha}, \sigma_{fha}^2\right) \mathcal{N}\left(s; \mu_d, \sigma_d^2\right) ds. \end{aligned}$$

This integration can be analytically computed by repeatedly applying the closed formula for the product of two Gaussians which is included for completeness as (6)

in the Appendix. To apply this operator we also re-express the $\mathcal{N}\left(s'; \zeta_{ha} s + \beta_{ha}, \sigma^2_{ha}\right)$ as a function of s by manipulating the exponent. Performing these operations, then integrating, yields the final expression for the new belief:

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

where

$$\begin{aligned}
C_1 &= \frac{\sigma^2_{fs}\sigma^2_d}{\left(\sigma^2_{fa} + \sigma^2\right) d} \\
c_1 &= \frac{\mu_d \sigma^2_{fa} + \mu_{fa}\sigma^2_d}{\sigma^2_{fa} + \sigma^2_d} \\
C_2 &= C_1 + \frac{\sigma^2_{zl}}{\zeta_{ha}} \\
c_2 &= \beta_{ha} + \zeta_{ha} c_1 \\
\sigma^2_{dfhl} &= \left(\left(\sigma^2_{zl}\right)^{-1} + C_2^{-1}\right)^{-1} \\
\mu_{dfhl} &= \sigma^2_{dfhl}\left(\mu_{zl}\left(\sigma^2_{zl}\right)^{-1} + c_2 C_2^{-1}\right) \\
c_{dfhl} &\propto w_d w_{fha} w_d \mathcal{N}\left(\mu_{zl}|c_2, \sigma^2_{zl} + C_2\right).
\end{aligned}$$

The weights are scaled so the belief is normalized such that $\int_s b^{a,z}(s)ds = 1$. Hence the representation of the belief as a mixture of Gaussians is closed under belief updating.

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, the Bellman equations can be expressed as a slight modification to (1), replacing sums with integrals and writing out the expectation:

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

where we have used the inner-product operator $\langle f, b \rangle$ as shorthand for expectation to obtain the second equality. As stated previously, α_{azj} is the α-function for the conditional policy corresponding to taking action a, receiving observation z and then following the previous $(t-1)$-step policy tree α_j, and can 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 parametric models yields

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

We combine the three Gaussians inside the integrand into a single Gaussian by repeatedly using the formula for the product of two Gaussians (6). This creates a function of s' and other terms that are independent of s'. Integrating over s' yields

$$\alpha_{azj} = \sum_{f=1}^{F} \sum_{h=1}^{H} \sum_{k=1}^{K} \sum_{l=1}^{L} w_{fhkl} \mathcal{N}\left(s;\mu_{fhkl},\sigma_{fhkl}^2\right) \tag{4}$$

where

$$\begin{aligned} w_{fhkl} &= w_{fha} w_k w_l \mathcal{N}\left(s_l;s_k,\sigma_k^2+\sigma_l^2\right) \mathcal{N}\left(\mu_{fha};\frac{c-\beta_{ha}}{\gamma_{ha}},\sigma_{fha}^2+\frac{C+\sigma_{ha}^2}{\gamma_{ha}^2}\right), \\ C &= \frac{\sigma_l^2\sigma_k^2}{\sigma_l^2-\sigma_k^2}, \\ \mu_{fhkl} &= \frac{\left(C+\sigma_{ha}^2\right)\mu_{fha}+\sigma_{fha}^2\gamma_{ha}\left(c-\beta_{ha}\right)}{\sigma_{fha}^2\left(C+\sigma_{ha}^2\right)}, \\ \sigma_{fhkl}^2 &= \frac{\sigma_{fha}^2\left(C+\sigma_{ha}^2\right)}{C+\sigma_h^2+\sigma_{fha}^2\gamma_{ha}^2}, \\ c &= \frac{\mu_k\sigma_l^2+\sigma_k^2\mu_l}{\sigma_l^2-\sigma_k^2}. \end{aligned}$$

The new α_{azj} now has $F \times H \times K \times L$ components, compared to the α_j for the $(t-1)$-step policy tree, which only had K components. To finish the value function

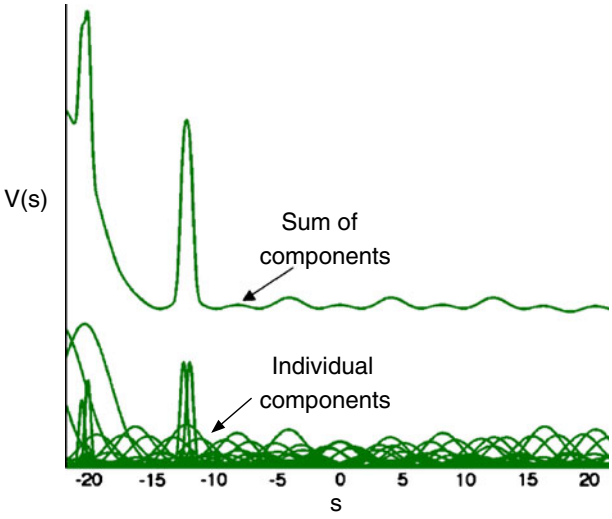


Fig. 4 A sample α-function, showing both its total value at each state, as well as the individual Gaussian components that are summed to create its final value

backup, we substitute α_{azj} back into (4) and choose the α-function that maximizes the future expected reward $\langle \alpha, b \rangle$ of belief b

$$\alpha(s) = \max_a R(s,a) + \gamma \sum_{z=1}^{|Z|} \max_{\alpha_{azj}} \alpha_{azj}$$

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

Since all elements in this result are weighted sums of Gaussians, the α-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 perform the integrals in closed form. Figure 4 displays an example value function, showing both its Gaussian components and its final (summed) representation.

Observe that the number of Gaussian components in a single α-function has increased: from K to $G + |Z|FHKL$ components.

4.2 Computational complexity of exact value function backups

Before proceeding, it is important to consider the computational complexity of an exact value function backup for a particular belief state. As just described, the first step is to create the set of new α_{azj} functions which consist of considering each possible action, then receiving each possible observation, and then following one of the initial α-functions, α_j. From (4) we know that creating each new α_{azj} components, involves $F \times H \times K \times L$ matrix operations (including a matrix inverse from combining two Gaussians, an $O(D^3)$ computation where D is the number of state dimensions D). This procedure is repeated for each of the $N_\alpha |A||Z|$ new α_{azj}

functions, yielding a cost of $O(FHKLN_\alpha|A||Z|D^3)$. It is then necessary to compute the expected value of the current belief state under each of these new functions in order to select the α_{azj} which maximizes the expected future reward of the current belief state under each possible action and observation. This operation involves the integral of the product of two weighted sums of Gaussians, an operation that can be done analytically and whose computational complexity is $O(D^3)$ for each of the combined Gaussians (to perform the necessary matrix inverse) multiplied by the number of components in each original expression. This operation will take $O(FHKLN_\alpha|A||Z|O(D^3)N_B)$ time.

Completing the backup requires summing over all observations for each action which is a simple summation over all the previously computed $\langle \alpha_{azj}, b \rangle$ values. Finally the immediate reward is added in which simply adds in G more Gaussian components. To select the best action requires also estimating the expected immediate reward, an $O(GD^3N_B)$ calculation for each action. In total the computational complexity for a single backup is $O((FHKLN_\alpha|Z| + G)|A|D^3N_B)$.

However, as multiple backups are performed, K, which represents the number of components used to represent each α-function, will increase by a factor of $FHL|Z|$ after each backup.[4] Therefore, when performing backups at horizon T, the α-functions will have about $K(FHL|Z|)^T$ components, and the computational cost of a single further backup will be $O(((FHL|Z|)^T KN_\alpha|Z| + G)|A|D^3N_B)$. This means that the computational cost of performing a single backup will grow exponentially with the time step of the backup performed.

4.3 Approximating the α-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 α-functions. Since α-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.[5] In Porta et al. [15]'s work they faced a similar (though smaller) expansion in the number of components. In order to reduce the number of components used to represent the belief states and α-functions, they used a slight variant of Goldberger and Roweis's method [7] that minimizes the Kullback-Leibler (KL) distance between an original model $f(x)$ and the approximation $\tilde{f}(x)$

$$D_{KL}\left(f||\tilde{f}\right) = \int_x f(x) \log \frac{f(x)}{\tilde{f}(x)} dx \tag{5}$$

where f and $\tilde{f}$ are weighted mixtures of Gaussians. However, the KL distance is not particularly appropriate as a distance measure for the α-functions since they are

[4] To be precise, it will the new number of components is $FHL|Z|K + G$ but the main factor is due to the multiplication with $FHL|Z|$.

[5] 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.

not probability distributions. In addition, the KL divergence 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.

We consider three alternate projection operators for computing an approximation of the α-function: highest peaks, minimizing the L2 norm, and minimizing the empirical max norm.

4.4 Highest peaks projection

A simple projection operator is to keep the N_C components with the highest peaks and discard all other components. The peak value of the i-th Gaussian component is simply

$$\frac{w_i}{(2\pi)^{D/2}|\Sigma_i|^{1/2}|}$$

where D is the state–space dimension, Σ_i is the variance of the i-th Gaussian, and w_i is its weight. It is simple and fast to compute all these values, and select the N_C largest components. Figure 5 shows an example of an original α-function, and the approximations obtained by keeping different numbers of components.

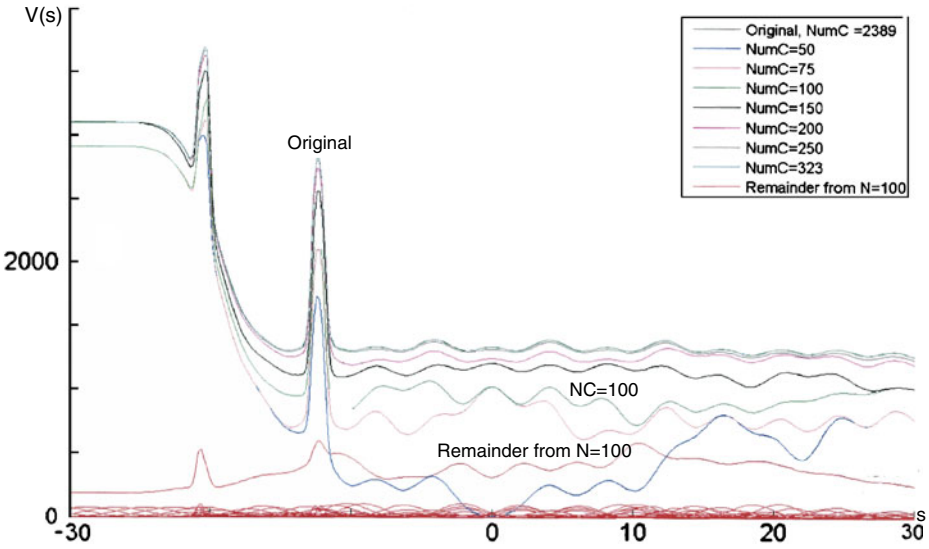


Fig. 5 Highest peak projection. Here the Gaussian components composing the original function are sorted by their maximum value and the highest N_C of these are kept. This figure displays the resulting approximations as more and more components are kept, along with the resulting residual for when $N_C = 100$. Though this is a simple approximation to compute, it is clear from the figure that the resulting approximation can be quite poor when N_C is small

4.5 L2 minimization

Another alternate distance measure without the shortcomings of the KL-divergence 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

$$\begin{aligned}
&\int_s \left[\sum_i^{N_C} w_i \mathcal{N}\left(s; \mu_i, \sigma_i^2\right) - \sum_j^{N} w_j \mathcal{N}\left(s; \mu_j, \sigma_j^2\right) \right]^2 ds \\
&\quad = \sum_i^{N_C} \sum_{i'}^{N_C} w_i w_i' \mathcal{N}\left(\mu_{i'}; \mu_i, \sigma_i^2 + \sigma_{i'}^2\right) + \sum_j^{N} \sum_j^{'N} w_j w_j' \mathcal{N}\left(\mu_{j'}; \mu_j, \sigma_j^2 + \sigma_{j'}^2\right) \\
&\qquad - 2 \sum_j^{N} \sum_i^{N_C} w_j w_i \mathcal{N}\left(\mu_i; \mu_j, \sigma_j^2 + \sigma_i^2\right).
\end{aligned}$$

While there is no analytic solution for the parameters w_i, μ_i, σ_i^2 that minimizes this expression, we can find an approximation to the optimal solution using Zhang and Kwok's recent work on reducing the number components in kernel function mixture models [22]. 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:

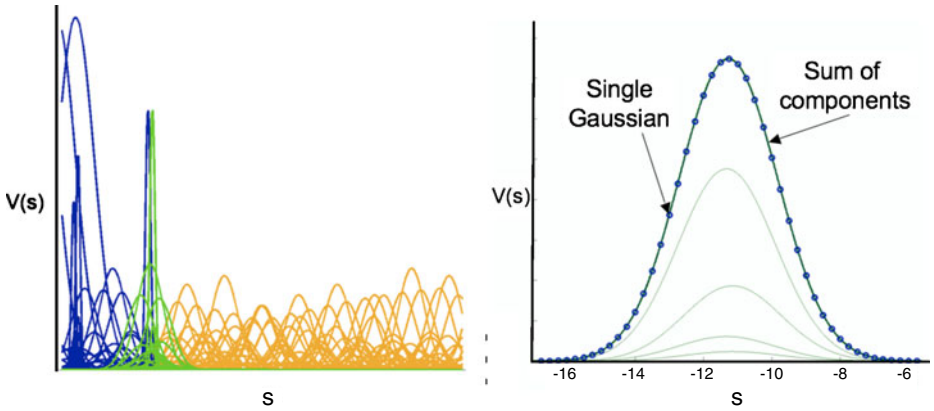
$$L2 \leq N_C \sum_{i=1}^{N_C} \int \left[w_i \mathcal{N}\left(s; \mu_i, \sigma_i^2\right) - \sum_{j \in S_i} w_j \mathcal{N}\left(s; \mu_j, \sigma_j^2\right) \right]^2 ds$$

where S_i is the i-th cluster and N_C is the total number of components in the approximation $\tilde{\alpha}$. The parameter fitting procedure is simplified 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 (see Fig. 6).

Though we can use this procedure to optimize an upper bound to the L2 norm, we would also like to constrain the exact L2 norm error to be within some threshold. To do this, we can formulate the 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, N_{Cmax}. This can be stated mathematically as follows:

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

where t is L2 norm threshold.



(a) Assign clusters (b) Fit cluster

Fig. 6 L2 clustering for approximating the α-functions. We iteratively cluster components and assign them to one of the components in the approximate α-function (see (**a**)). Then given all the components assigned to the same cluster, we fit a single Gaussian that approximates the sum of all components in the same cluster: this second stage is depicted in (**b**) for a particular set of components. Note that in (**b**) the line representing the sum of all components is visually identical to the single Gaussian (shown by the *dotted line*) used to approximate the sum of components. This process is then iterated until a convergence criteria is met

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 N_C in the approximation is increased until either an initialization is found that lies within the feasible region of the constraints, or N_C no longer satisfies the second constraint. If both constraints cannot be satisfied, N_C is set to N_{Cmax}.

Once the parameters are initialized and M is fixed, we follow Zhang and Kwok's procedure for optimizing. Experimentally this approach was found to produce decent results. Fig. 7 displays an example fit where though the L2 norm is still fairly large, the original and approximate α-functions are indistinguishable to the naked eye.

4.6 Point-based minimization

One of the limitations of the L2-based approximation method presented is that it may be computationally too slow in practice. An alternate choice is a faster heuristic method which greedily minimizes the max norm at a subset of points along the value function.

In this method, the α-function is first sampled at regular intervals along the state space. These sampled values are considered potential residuals r. First the largest magnitude residual (either positive or negative) is found and a Gaussian is placed

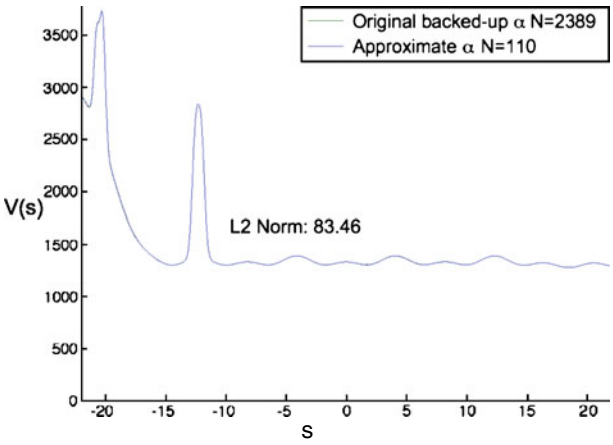
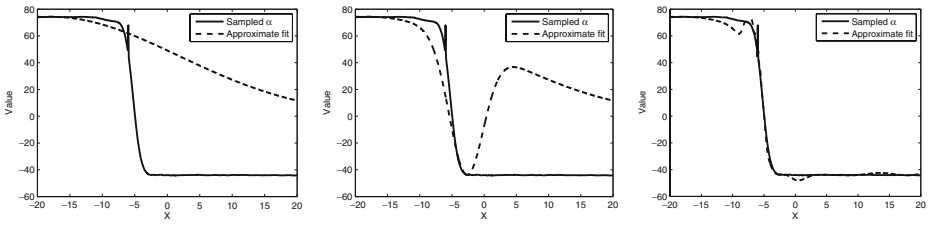


Fig. 7 The original (*in green*) and approximated α-function (*in blue*). The original α-function was composed of 2,389 components. The approximate α only uses 110 components, yet in the plotted function is indistinguishable from the original α

at this point, with a height equal to the residual and variance approximated by estimating the nearby slope of the function. This Gaussian is added to the set of new components. Then a new set of residuals is computed by estimating this Gaussian along the same intervals of the state space, and subtracting its value from the original residuals: $r = r - w_i \mathcal{N}(s|\mu_i, \Sigma_i)$. This creates a new set of residuals. We now find the maximum magnitude sample of this set of residuals and fit a second Gaussian in the same manner as before. This process is repeated until the number of new Gaussians reaches the fixed size of our representation for the α-functions. See Fig. 8 for an illustration of this process. Note that at each round during this process we are adding a Gaussian component at the location which currently has the max norm error at the sampled points. Therefore this algorithm takes a greedy approach to minimizing the max norm between the approximate $\tilde{\alpha}$-function and the original α-function along the set of sampled points.

4.7 Planning

We now have the major components necessary to apply a point-based approach to POMDP planning: specifically, a belief update operator, a value function backup



(a) 1 component (b) 2 components (c) 15 components

Fig. 8 Approximating the α-functions. We use a simple residual fitting technique. First the function is sampled. Then a Gaussian component is added to reduce the highest component. This process is repeated until the maximum number of Gaussians is reached

procedure, and a projection operator to keep the number of parameters used to represent the value function bounded. Algorithm 1 outlines the SM-POMDP algorithm. Inspired by results in discrete-state planners [16, 17] we create a belief set B by sampling belief states in a set of short trajectories, though in our case we simply perform random actions from an initial starting belief state b_0. We initialize the value function as a single α-function in a domain-dependent manner. Starting with this initial α-function, we iteratively perform value function backups for the chosen belief set B. At each round, we select a trajectory from our belief set and back up the beliefs in it in reverse order. In problems where a number of steps must be executed in order to reach a high reward state this approach was found to be superior to randomly backing up belief points.

Algorithm 1 SM-POMDP

Input: N_{traj} (number of trajectories), N_T (number of beliefs per trajectory), POMDP
for i=1:N_{traj} **do** {Generate belief set B}
  $b = b_0$
  Draw an initial state s from b_0
  for j=1:N_T **do**
    Select an action a
    Sample $s' \sim p(s'|s, a)$
    Sample an observation $z \sim p(z|s')$
    $b^{a,z} = BeliefUpdate(b, a, z)$
    Add belief to set: $B[j, i].b = b^{a,z}$
    $b = b^{a,z}, s = s'$
  end for
end for
Initialize value function
loop {Perform planning until termination condition}
  Select a trajectory $r \in 1 : N_{traj}$ at random
  for t=T:-1:1 **do**
    Set V_{old} to current value of t-th belief $B[r, t].b$
    α_{new} = ValueFunctionBackup(B[r,t].b)
    Project to smaller rep: $\alpha'_{new} = ProjectDownToN_CComponents(\alpha_{new})$
    if Projected value still improved ($\langle \alpha'_{new}, B[r, t].b \rangle > V_{old}$) **then**
      Add α_{new} to value function
    end if
  end for
end loop

5 Analysis

We next provide bounds on a subset of the SM-POMDP variants' performance, and also analyze the computational complexity of all the prior SM-POMDP variants discussed.

In terms of performance quality, our objective lies in bounding the potential difference between the optimal value function, denoted by V^* and the value function computed by the SM-POMDP algorithm, denoted by V^{SM}. Let H represent an exact value function backup operation, H^{PB} a point-based backup operation, and A the projection operator which maps the value function to a value function with a smaller number of components. Our goal is a bound on the error between the n-step optimal value function V_n^* and the SM-POMDP value function V_n^{SM}. The total error $||V^* - V_n^{SM}||_\infty$ is bounded by the n-step error plus the error between the optimal n-step and infinite horizon value functions, $||V^* - V_n^*||_\infty$ which is bounded by $\gamma^n||V^* - V_0^*||_\infty$. We will show the n-step error can be decomposed into a term due to the error from point-based planning and a term due to the error from the projection to a fixed number of components.[6]

Theorem 1 *Let the error introduced by performing a point-based backup instead of an exact value backup be $\epsilon_{PB} = ||HV - H^{PB}V||_\infty$ and let the error introduced by projecting the value function down to a smaller number of components be $\epsilon_{\text{proj}} = ||V - AV||_\infty$. Then the error between the optimal n-step value function V_n^* and the value function computed by the SM-POMDP algorithm is bounded by*

$$\left\| V_n^* - V_n^{SM} \right\|_\infty \leq \frac{\epsilon_{PB} + \epsilon_{\text{proj}}}{1 - \gamma}.$$

Proof We proceed by first re-expressing the error in terms of the backup operators, and then add and subtract a new term in order to separate out the error due to performing backups on different value functions, versus the different backup operators themselves:

$$\begin{aligned}
\left\| V_n^* - V_n^{SM} \right\|_\infty &= \left\| HV_{n-1}^* - AH^{PB}V_{n-1}^{SM} \right\| \\
&\leq \left\| HV_{n-1}^* - HV_{n-1}^{SM} \right\|_\infty + \left\| HV_{n-1}^{SM} - AH^{PB}V_{n-1}^{SM} \right\|_\infty \\
&\leq \gamma \left\| V_{n-1}^* - V_{n-1}^{SM} \right\|_\infty + \left\| HV_{n-1}^{SM} - AH^{PB}V_{n-1}^{SM} \right\|_\infty
\end{aligned}$$

where the result follows by the use of the triangle inequality.

We then add and subtract $H^{PB}V_{n-1}^{SM}$ in order to separate the error introduced by performing a point-based backup, from the error introduced from the projection operator, and again use the triangle inequality:

$$\begin{aligned}
\left\| V_n^* - V_n^{SM} \right\|_\infty &\leq \gamma \left\| V_{n-1}^* - V_{n-1}^{SM} \right\|_\infty + \left\| HV_{n-1}^{SM} - H^{PB}V_{n-1}^{SM} \right\|_\infty + \dots \\
&\quad + \left\| H^{PB}V_{n-1}^{SM} - AH^{PB}V_{n-1}^{SM} \right\|_\infty \\
&\leq \gamma \left\| V_{n-1}^* - V_{n-1}^{SM} \right\|_\infty + \epsilon_{PB} + \epsilon_{\text{proj}}.
\end{aligned}$$

[6] In this analysis we ignore possible error due to the approximate normalization of the observation and mode models, assuming that enough Gaussians are used to represent these functions that they can be made to sum arbitrarily close to 1 and therefore contribute negligible error.

This is a geometric series, therefore

$$\left\| V_n^* - V_n^{SM} \right\|_\infty \leq \frac{\left(1-\gamma^{n+1}\right)\left(\epsilon_{PB}+\epsilon_{\text{proj}}\right)}{1-\gamma}$$
$$\leq \frac{\epsilon_{PB}+\epsilon_{\text{proj}}}{1-\gamma}.$$

□

In the following sections we will discuss bounds on ϵ_{proj}, but the remainder of this section will focus on ϵ_{PB}.

Pineau et al. [14] previously provided a proof for bounding the error induced by performing a point-based backup, ϵ_{PB} for discrete-state POMDPs. We now provide a similar proof for continuous-state POMDPs. $d_{KL}(p,q)$ will denote the Kullback-Leibler divergence between two probability distributions p and q. B is the set of points in the belief set used to perform point-based backups and Δ is the full belief space.

Lemma 1 *Let $w_{\min}$ be the lowest weight and $w_{\max}$ be the highest weight of the weights used to specify the original weighted sum of Gaussians reward function. Let $|\Sigma_{\min}|$ be the smallest determinant of the variance of the Gaussians, and let N_C be the maximum number of components used to represent the weighted sum of Gaussians. Then the error introduced by performing a point-based backup is at most $\epsilon_{PB} = \frac{N_C(w_{\max}-w_{\min})\max_{b'\in\Delta}\min_{b\in B} 2\sqrt{2d_{KL}(b,b')}}{(1-\gamma)(2\pi)^{D/2}|\Sigma_{\min}|^{1/2}}$.*

Proof Following Pineau et al. [14] let $b' \in \Delta$ be the belief at which the point-based value function has its largest error compared to the full backed up value function, and let $b \in B$ be the closest sampled point in the belief set B to b' where distance is defined in terms of the KL-divergence. Let α' and α be the two functions which respectively maximize the expected value of b' and b. By not including α' the point-based operation makes an error of at most $\int_s \alpha'(s)b'(s) - \alpha(s)b'(s)ds$.

To bound this error, we first add and subtract the same term:

$$\epsilon_{PB} \leq \int_s \alpha'(s)b(s) - \alpha(s)b'(s)ds$$
$$= \int_s \alpha'(s)b(s) - \alpha(s)b(s) + \alpha(s)b(s) - \alpha(s)b'(s)ds.$$

The value of α' at b must be lower than α since α is defined to give the maximal value. Therefore

$$\epsilon_{PB} \leq \int_s \alpha'(s)b(s) - \alpha'(s)b(s) + \alpha(s)b(s) - \alpha(s)b'(s)ds$$
$$= \int_s \left(\alpha'(s) - \alpha(s)\right)\left(b(s) - b'(s)\right)ds$$
$$\leq \int_s \left|\left(\alpha'(s) - \alpha(s)\right)\left(b(s) - b'(s)\right)\right|ds.$$

Hölder's inequality can be used to bound this expression as follows

$$\epsilon_{PB} \leq \left(\int_s |\alpha'(s) - \alpha(s)|^{\infty} ds \right)^{1/\infty} \left(\int_s |b(s) - b'(s)| ds \right)$$

where the first term is the max norm of $\alpha' - \alpha$ and the second term is the 1-norm between b and b'. Since there is not an easy way to estimate the 1-norm between two Gaussians or weighted sums of Gaussians, we now relate this 1-norm to the KL-divergence using the results from Kullback [10] (included for completeness as Lemma 2 in the Appendix):

$$\epsilon_{PB} \leq \|\alpha'(s) - \alpha(s)\|_{\infty} 2\sqrt{2 d_{KL}(b, b')}.$$

The worst error in immediate reward between two value functions occurs when all N_C components have the same mean and the smallest possible variance $\Sigma_{\min}$, and where one value function has all weights set at $w_{\max}$ and the other value function has all weights set at $w_{\min}$. This yields an error of $\frac{N_C(w_{\max} - w_{\min})}{(2\pi)^{D/2}|\Sigma_{\min}|^{1/2}}$. The infinite horizon values in this worst case would magnify these errors by at most $1/(1-\gamma)$. Therefore the max norm error between two value functions is bounded by

$$||\alpha' - \alpha||_{\infty} \leq \frac{N_C(w_{\max} - w_{\min})}{(1-\gamma)(2\pi)^{D/2}|\Sigma_{\min}|^{1/2}}$$

and so the full point-based error is bounded as follows:

$$\epsilon_{PB} \leq \frac{N_C(w_{\max} - w_{\min}) 2\sqrt{2 d_{KL}(b, b')}}{(1-\gamma)(2\pi)^{D/2}|\Sigma_{\min}|^{1/2}}.$$

□

Belief points can be chosen to reduce this error, such as in Pineau's PBVI [14] algorithm which adds belief points that have the largest error (which in their proof is measured in by the 1-norm) along some trajectory relative to the existing beliefs in B. One slight complication is how to estimate the KL-divergence between the belief points. If the belief points are represented as single Gaussians then there exists a closed form expression for the KL-divergence between two beliefs. There exists no general closed form representation between the KL-divergence between two weighted sums of Gaussians. However, a recent paper by Hershey and Olsen [8] investigates a number of approximations to this quantity, including an upper bound that is fast to compute, which can be utilized to ensure that the error estimated is an upper bound to the true error.

The second cause of error in the final value function is due to the projection step, ϵ_{proj}, and will depend on the particular projection operator used. We next consider four different operator choices and discuss where possible their associated ϵ_{proj}. We also consider the computational complexity of the resulting SM-POMDP algorithm under each of these choices, considering the cost of the projection operator and the backup operator. The results of this section are summarized in Table 1.

Table 1 Summary of the estimated error introduced by the different SM-POMDP projection operators, and the computational complexity of using each operator as part of a POMDP planner

Projection algorithm	ϵ_{proj}	Computational complexity of the T-th step		Empirical performance
		Projection operator	Backup operator	
No projection	0	0	$O(((FHL\|Z\|)^T K N_\alpha \|Z\| + G)\|A\| D^3 N_B)$	Computationally intractable
Highest peak	$\frac{(FHL\|Z\|N_C+G-M)\max_j w_j}{(2\pi)^{D/2}\min_j \|\Sigma_j\|^{1/2}}$	$O((FHL\|Z\|N_C + G)D^3 + (FHL\|Z\|N_C + G)^2)$	$O((FHL\|Z\|N_C N_\alpha \|Z\| + G)\|A\| D^3 N_B)$	Poor quality approximation
L2 minimization	Computes ϵ_{L2} which is not guaranteed to be an upper bound to ϵ_{proj}. However it tries to fit a good approximation to the value function over the entire state space.	$O(N_C(FHL\|Z\|N_C + G)D^3)$	$O((FHL\|Z\|N_C N_\alpha + G)\|A\| D^3 N_B)$	Good quality approximation but slower than point-based
Point-based	$\lim_{Q\to\infty} \tilde{\epsilon}_{\text{proj}} \to \epsilon_{\text{proj}}$. The empirical norm $\tilde{\epsilon}_{\text{proj}}$ can quickly become small using only a few components.	$O(Q_1^D(FHL\|Z\|N_C + G)D^3)$	$O((FHL\|Z\|N_C N_\alpha + G)\|A\| D^3 N_B)$	Good quality approximation and faster than L2 minimization in low-dimensional domains used

D is the number of state dimensions, $|A|$ is the number of actions, $|Z|$ is the number of observations, N_B is the number of components in the belief state, K is the number of components in the original α-functions, and the remaining symbols are defined in the text

5.1 Exact value function backups

If the value function backups are performed exactly then ϵ_{proj} will equal zero. Therefore exact value backups offer some appealing performance guarantees. Unfortunately, as discussed in Section 4.2, there is a significant computational cost to these guarantees. This cost will often become infeasible when the backup is iteratively performed several times and F, H, L, or $|Z|$ are large.

We next consider the three projection operators.

5.2 Highest peak projection analysis

Take the projection method described in Section 4.4 and assume that prior to the backup each α-function consists of N_C Gaussians. Then the backup procedure creates a new α with $FHL|Z|N_C + G$ components. Therefore, assuming the Gaussians are sorted in descending order of peak value, the error introduced by projecting to N_C components is the value of the discarded $FHL|Z|N_C + G - N_C$ Gaussians,

$$\epsilon_{\text{proj}} = \sum_{j=N_C+1}^{FHL|Z|N_C+G} w_j \mathcal{N}\left(s|\mu_j, \Sigma_j\right).$$

In the worst case all of these Gaussians would be centered at the same location, yielding an error of

$$\epsilon_{\text{proj}} \leq \frac{\left(FHL|Z|N_C + G - N_C\right) \max_j w_j}{(2\pi)^{D/2} \min_j |\Sigma_j|^{1/2}}.$$

This provides an upper bound on the projection error which, in turn, yields a bound on the overall error between the SM-POMDP value function and the optimal value function. In addition, the computational cost of this procedure is fairly cheap. It requires calculating the determinant of each component, which is an $O(D^3)$ operation where D is the number of state dimensions, and then a sorting algorithm to identify the N_C components with the largest peak. Quicksort or a similar algorithm can be used which has a worst case complexity of the number of items squared. Therefore the computational complexity of a highest peak projection is

$$O\left((FHL|Z|N_C + G)D^3 + (FHL|Z|N_C + G)^2\right).$$

However, despite the appeal of formal theoretical guarantees, unless a large number of components is used, the empirical quality of the approximation may not be as good as the other two projection methods (as is illustrated by comparing Fig. 5 to Figs. 7 and 8).

5.3 L2 minimization analysis

Section 4.5 outlined a projection method where the objective is to minimize the L2 norm between the original and projected value function. A strength of this approach is that it is possible to analytically evaluate the final L2 norm. However, in continuous spaces L2 is not necessarily an upper bound to the max norm. This can be seen by

considering the case of a delta function: the L2 norm of this function is zero, but the max norm is infinity. Though this approach does not provide a direct estimate of ϵ_{proj}, it does try to closely approximate the value function over the entire state space, rather than focusing on the worst error.

The algorithm presented in Section 4.5 is an iterative procedure. It requires repeatedly computing the L2 norm between two weighted sets of Gaussians, one with $FHL|Z|N_C + G$ components, and one with N_C components. This involves computing the determinant of D-dimensional matrices, and so the full computational complexity of this projection operator is $O((FHL|Z|N_C + G)MD^3)$. This calculation is repeated many times during the procedure.

5.4 Point-based minimization analysis

In Section 4.6 we discussed an approach that greedily minimizes the empirical max norm of a set of Q points sampled along the value function. Let $\tilde{\epsilon}_{\text{proj}}$ represent the empirical max norm computed. As the number of points sampled goes to infinity, this error will converge to the true projection error ϵ_{proj}:

$$\lim_{Q \to \infty} \tilde{\epsilon}_{\text{proj}} = \epsilon_{\text{proj}}.$$

Unfortunately we do not know of any finite Q guarantees on the relationship between the empirical max norm and the true max norm error. However, in practice, a small max norm error on the sampled points often indicates that there is a small max norm between α and $\tilde{\alpha}$.

This method requires computing the value function at a set of Q points which is an $O(Q(FHL|Z|N_C + G)D^3)$ operation due to the need to compute the inverse and determinant of the Gaussian variances. However, the number of points Q necessary to compute a good estimate of the value function will increase exponentially with the state dimension: essentially this procedure requires computing a discretized estimate of the value function, and then refits this estimate with a new function. If Q_1 is the number of points used to estimate a one-dimensional value function, then Q_1^D will be the number of points needed to estimate a D-dimensional function. Therefore we can re-express the computational complexity as

$$O\big(Q_1^D(FHL|Z|N_C + G)D^3\big).$$

Generally this projection operator will work best in low dimensional domains.

Before summarizing these approaches in Table 1, we need to also briefly provide an analysis of the computational complexity of performing backups when the value function is projected to a smaller number of components.

5.5 Computational complexity of the backup operator when using a projection operator

The computational cost analysis is similar to the exact value function case presented in Section 4.2, with one important alteration: the number of components in the resulting α-functions is now restricted. As before, let N_C be the maximum number of components in an estimated α-function during planning. N_C can either be specified in advance, or computed during planning if the number of components is dynamically adjusted in order to prevent the approximation step from exceeding a maximum

desired error. In this case the computational complexity of performing a single backup never exceeds $O(FHLN_C N_\alpha |A||Z|D^3 N_B)$, regardless of the horizon at which the backup is performed (or in other words, regardless of the number of prior backups). This means that the computational cost is bounded (assuming that N_C does not grow as a direct function of the horizon T). The choice of N_C provides the above mentioned parameter for tuning between the quality of the resulting solution and the computational cost required to compute that solution. In our experiments we show it is possible to achieve good experimental results by fixing the value of N_C in advance.

5.6 Analysis summary

Table 1 summarizes the computational complexity of performing a value function backup, and the additional error and computational overhead introduced by projecting the resulting α-function down to a smaller number of components. In general, it will be computationally necessary to perform a projection if more than a couple value function backups are required (that is, if T is greater than 2 or 3). To provide some intuition for this, we provide a simple example. Assume the following problem setup:

- There are 2 observations ($|Z|| = 2$) each represented by 5 components ($L = 5$)
- There are 2 actions ($|A| = 2$) each with 3 modes ($H = 3$) and each mode is represented using 5 components ($F = 5$)
- Each reward is represented with 3 components ($G = 3$)
- $N_\alpha = 1$
- The original α-function is represented as a re-scaled reward function with 3 components ($K = 3$)
- $N_B = 1$
- $N_C = 50$

Then the number of components after a single point-based backup is

$$FHL|Z|K + G = 5 * 3 * 5 * 2 * 3 + 3 = 453.$$

The number of components after the second exact point-based backup can then be

$$\begin{aligned} FHL|Z|453 + G &= 5 * 3 * 5 * 2 * 453 + 3 = 67{,}503 \\ &= (FHL|Z|)^2 K + FHL|Z|KG + G^2, \end{aligned}$$

and on the third round

$$FHL|Z|67{,}503 + G = 10{,}125{,}453 \approx (FHL|Z|)^3 K.$$

In contrast, if the α-function is projected to N_C components after each backup, the number of new components created during a backup is fixed at

$$FHL|Z|N_C + G = 5 * 3 * 5 * 2 * 50 + 3 = 7{,}503.$$

Therefore, as long as the projection operator is cheaper than manipulating and updating a set of components that grows as $(FHL|Z|)^T$ where T is the number of backups, then projecting will be significantly faster than maintaining all components.

Empirically we found that the max-norm projection method was faster than the L2 norm projection method, and that both gave significantly higher quality results than the highest peaks method. Therefore we used the max-norm projection method in all experiments.

6 Experiments

The key advantage of SM-POMDP over prior techniques is its ability to handle a broader class of dynamics models. Therefore we examined its performance on three simulated examples that involve switching and/or multi-modal dynamics. The first example is a navigation domain where a robot must locate a hidden power supply. This problem exhibits switching dynamics because the robot's transitions depend on whether it is currently in the free space of the hallway, or near one of the walls. In the second example we consider a faulty robot trying to hop across treacherous terrain to reach a goal destination. This problem exhibits multi-modal dynamics since the robot's attempts to jump only sometimes succeed. In the final example we consider a two-dimensional unmanned aerial vehicle simulation where one agent must avoid another agent.

Except where otherwise noted, we focus on comparing SM-POMDP to a linear-model parametric continuous-state planner, such as the planner developed by Porta et al. [15]. This is the most similar planner to SM-POMDP as only the dynamics models are changed, and is therefore an important benchmark.

In all examples SM-POMDP outperformed the comparison linear-model continuous-state planner. α-functions were projected down to at most 50 components using the empirical max norm projection method. Beliefs were restricted to have four components, and were projected down using the KL-divergence technique from Goldberger and Roweis [7] which is more appropriate for beliefs than value functions since beliefs represent a probability distribution over states.

Porta et al. had previously demonstrated that their planner was faster than a standard discrete-state planner Perseus [20] on a simulation problem, demonstrating the benefits of parametric planners that automatically adjust the parameterization of the value function during planning. In the first experiment we confirm that this advantage holds for SM-POMDP which can handle domain dynamics that are poorly approximated by a single linear Gaussian model. In addition we show this advantage when comparing to a state-of-the-art discrete-state planner (HSVI2 [18]) which is known to outperform Perseus [20]. We then focus our experimental efforts on demonstrating the additional flexibility and modeling power of SM-POMDP that allows it to find better plans than linear Gaussian POMDP planners.

6.1 Finding power: variable resolution example

In the first experiment 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. This causes the dynamics to exhibit switching:

if the robot is in the middle of the hallway it will have different dynamics than near the walls. 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 reward of 12.6 (modeled by a highly peaked Gaussian); otherwise it receives a lesser reward of 5.8. Therefore it is least preferable to move around, and it is best to plug in at the correct location. The power supply lies beneath the robot sensors so the robot is effectively blind. However, if the robot knows the true world dynamics, it can localize itself by travelling to one end of the hallway, reasoning that it will eventually reach a wall.

In addition to the switching dynamics, this problem is also interesting because the resulting value function is smooth for large parts of the state space, intermixed with small sections that are rapidly changing. Therefore is another demonstration of where continuous-state parametric planners' dynamically adjustable value function representation may have an advantage over discrete-state planners. To illustrate this, we compare to HSVI2 [18], a highly optimized, state-of-the-art discrete-state POMDP planner.

We present results for the discrete-state HSVI2 planner, a linear continuous-state planner and SM-POMDP. The two continuous-state planners were run on 100 trajectories. Each trajectory was gathered by starting in a random state $s \in [-21, 21]$ with a Gaussian approximately uniform belief and acting randomly for episodes of 10 steps. This lead to a total set of 1000 belief states. 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.

All algorithms require a stopping threshold. We set our continuous-hybrid approach to do at least 10 rounds, and halt when $\sum_b V_t(b) - \sum_b V_{t-1}(b)| \leq 0.001$. HSVI2 halts when the upper and lower bounds on the value of the initial belief state falls below a certain threshold. The default setting of the HSVI2 stopping threshold is 0.001. We ran experiments using this value and also a higher value of 1.5 which corresponded roughly to a ratio measure we originally used.

The value functions produced by each planner were tested by computing the average reward received over 100 episodes of 50 steps/episode using the policy associated with the α-functions/vectors in the computed value function. At the start of each episode the robot was placed at a state s that was 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. The results are displayed in Table 2.

Table 2 Power supply experiment results

Models	Continuous-state		HSVI2: discretized number of states				
	Linear	SM-POMDP	840	1,050	1,155	1,260	2,100
Time(s) $\epsilon = 0.001$	30.6	548	577	28,906	86,828	23,262	75,165
Time(s) $\epsilon = 1.5$	n/a	n/a	–	9,897	34,120	5,935	23,342
Reward	290	465*	290	290	290	484*	484*

There was no significant difference between the rewards received by the good policies (marked by a *)

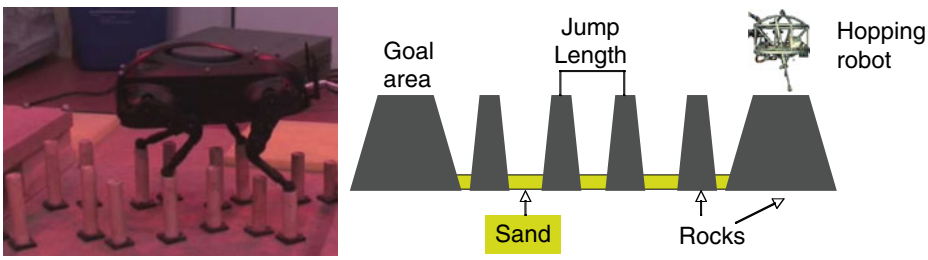
Our hybrid model found a good policy that involves taking big and small actions to first localize the belief (at the wall) and then taking three more steps to reach the power socket. The linear model continuous-state POMDP planner runs faster but fails to find a good policy since it cannot model the state-dependent dynamics near the wall edges, and there are no unique observations. Therefore the linear planner thinks localization is hopeless.

The discrete-state solutions do poorly at coarse granularities since the *PlugIn* reward gets washed out by averaging over the width of a too-wide discrete state. At fine state granularities the discrete approach finds a good policy but require more time. Our continuous-state planner finds a solution significantly faster than the coarsest discrete-state planner that can find a good solution. This result shows that Porta et al.'s previous demonstration of the benefits of parametric representations over discrete-state representations [15] are also present in domains with more complex dynamics.

6.2 Locomotion over rough terrain: bimodal dynamics

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 (see Fig. 9a). 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 (see Fig. 9b). 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 10 displays the multi-modal state-dependent nature of the dynamics. The robot is receives a penalty of -59 for stepping into the sand, and -1.1 for each step on the rocks. A *Signal* action results in a reward of ≈ 19.5 if the robot has reached the final location, or -0.1. All rewards are specified using a weighted sum of Gaussians. The observation model provides a noisy estimate of where the robot is (sand, rock 1–4, start or finish).

We tested SM-POMDP 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,



(a) Tricky navigation (b) Observation Model

Fig. 9 (**a**) Displays a motivating example for this experiment: navigation over a challenging set of poles by a quadruped robot (photo reproduced with permission from Byl and Tedrake [4]). (**b**) Shows the particular experimental domain used. A robot must try to reach the other large rock by hopping across the obstacles and then signalling it has reached the end

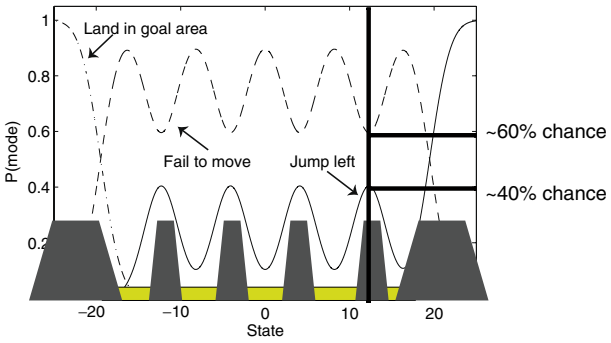


Fig. 10 Multi-modal state-dependent dynamics of *Jump*. A given mode's probability (such as *step left*) varies over the state space, and more than one mode is possible for a state

scaled to account for discounting and performing the action indefinitely. The results are displayed in the following table:

	Average total reward received
SM-POMDP	302.8
Linear model	−4.7

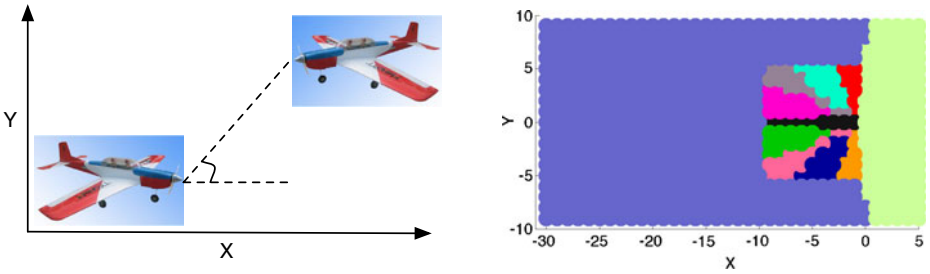
Since the hybrid model can correctly represent that a step from the initial platform will keep the robot on the platform or move it to the first rock, it can find a good policy of repeatedly trying to step and will eventually reach the goal platform. In contrast, the linear model performs poorly because 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.

A belief compression technique that assumes a unimodal belief state, such as Brooks et al. [2] approach, cannot perform optimally in this domain. This is because the belief state resulting from an action is inherently bimodal due to the faulty dynamics model, and a unimodal approach must, for example, average, or select one peak of the belief. When we evaluated our SM-POMDP approach with only a single component per belief, the resulting average reward was 255, significantly lower than the average reward of 302.8 received by SM-POMDP.

6.3 UAV avoidance

In our final experiment we investigated the performance of SM-POMDP on a two-dimensional simulated collision avoidance problem. The challenge is for the agent, an unmanned aerial vehicle, to successfully avoid crashing with an incoming aerial vehicle. The state is represented by the relative x and y between the two planes (see Fig. 11a[7]). The agent has 6 possible actions: it can go slowly forward ($x = +1$, $y = +1$), slowly up and forward ($x = +1$, $y = +0$), slowly down and forward ($x = +1$, $y = -1$), fast forward ($x = +5$, $y = +0$), fast forward and up ($x = +5$, $y = +1$), or fast forward and down ($x = +5$, $y = -1$). The incoming UAV always moves

[7]The airplane graphic in this figure is from http://www.germes-online.com/direct/dbimage/50200967/Radio_Controlled_Airplane.jpg.



(a) Plane Setup (b) Observation model

Fig. 11 UAV experiment. (**a**) Shows the basic problem setup: an agent must avoid an incoming UAV. (**b**) Shows the observation model. Different colors represent the placement of the centers of the Gaussians associated with different observations. When the agent is close to the incoming UAV they get a fairly high resolution estimate of the relative bearing to the incoming UAV, otherwise the agent receives only coarse observations

forward (from its perspective, so that it comes closer to the agent) but it can also stochastically choose between going straight, up and down. Therefore the dynamics model has three modes which are equally likely and depend on the (unknown) action of the incoming UAV. The agent incurs a small cost (of 3) for each forward action, a slightly larger cost (5) for going up or down and a large cost of 200 for getting within 1 unit of the incoming UAV. However, after the agent has passed the incoming UAV, it is rewarded by having the cost of actions drop to zero. The agent receives one of 11 discrete observations. If the agent is close to the incoming UAV, the agent receives a discretized bearing estimate to the incoming UAV. Otherwise the agent is too far away and receives a broad range estimate that essentially specifies whether the agent is in front or behind the other UAV. As with all the world models, these observation models are represented by a weighted set of Gaussians. The centers of those Gaussians are shown in Fig. 11b.

We compared SM-POMDP to the linear model POMDP planner. Both approaches used 1,000 belief points during planning. The resulting approaches were evaluated on 200 episodes, each of length 30 steps. The agent is always started between $(-30, -28)$ in the x direction, and $(-4, 4)$ in the y direction. The results are displayed in Table 3. This task is fairly easy, but SM-POMDP still outperforms the linear model. SM-POMDP yields a policy with a lower average cost per episode of 47.2, whereas the linear model incurs the greater cost of 50.5. The difference is statistically significant (t-test, $p < 0.001$). This improvement in performance is likely to be due to the fact that SM-POMDP has a better model of the incoming UAV's dynamics: in order to model the UAV as having a single dynamics model, the linear dynamics uses a dynamics model with a higher variance than is really present. This can cause the linear model to think that the incoming UAV could transition to some

Table 3 UAV avoidance average cost results, lower is better

Algorithm	Average cost/episode
SM-POMDP	47.2
Linear model	50.5

SPOMDP outperforms the linear model

states which the true model cannot reach, and therefore the linear model is overly conservative.

7 Conclusion

In this paper we introduced SM-POMDP, an algorithm for planning in continuous-state domains characterized by switching-state, multi-modal dynamics. The parametric representation of the dynamics model allowed all POMDP planning operators (belief updating and value function backups) to be performed in closed form. SM-POMDP can handle more general dynamics models than the prior work it builds on by Porta et al. [15] and the benefit of this was demonstrated in three simulated experiments. We also provided a formal analysis of SM-POMDP, bounding the resulting SM-POMDP value function relative to the optimal value function where possible, and analyzing the computational complexity of the SM-POMDP algorithm variants. SM-POMDP indicates the potential promise of parametric approaches in planning in partially-observable, continuous-state environments with complex dynamics.

Acknowledgements Emma Brunskill and Nicholas Roy were supported by the National Science Foundation Division of Information and Intelligent Systems under grant # 0546467. Emma Brunskill also gratefully acknowledges additional support from a Hugh Hampton Young Memorial Fund fellowship.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

Appendix

Formula for product of 2 Gaussians, as referenced in Porta et al. [15, p. 15]

$$\mathcal{N}(s|a, A)\mathcal{N}(s|b, B) = \mathcal{N}(a|b, A + B)\mathcal{N}(s|c, C) \tag{6}$$

where

$$C = \left(A^{-1} + B^{-1}\right)^{-1}$$
$$c = C\left(A^{-1}a + B^{-1}b\right)$$

Lemma 2 (Theorem from Kullback [10]) *Let p_1 and p_2 be two probability density functions defined over $\mathcal{X}$. Define*

$$\Omega = \left\{x \in \mathcal{X} \mid p_1(x) \geq p_2(x)\right\}.$$

If p_1 and p_2 are both measurable (integrable) over Ω, then

$$d_{KL}(p_1, p_2) \geq \frac{1}{8}||p_1 - p_2||_1^2$$

where d_{KL} is the Kullback-Leibler divergence.

References

1. Blackmore, L., Gil, S., Chung, S., Williams, B.: Model learning for switching linear systems with autonomous mode transitions. In: Proceedings of the IEEE Conference on Decision and Control (CDC) (2007)
2. Brooks, A., Makarenko, A., Williams, S., Durrant-Whyte, H.: Parametric POMDPs for planning in continuous state spaces. In: Robotics and Autonomous Systems (2006)
3. Burl, J.B.: Linear Optimal Control. Prentice Hall (1998)
4. Byl, K., Tedrake, R.: Dynamically diverse legged locomotion for rough terrain. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Video Submission (2009)
5. Fox, E.B., Sudderth, E.B., Jordan, M.I., Willsky, A.S.: Nonparametric Bayesian learning of switching linear dynamical systems. In: Advances in Neural Information Processing Systems (NIPS) (2009)
6. Ghahramani, Z., Hinton, G.: Variational learning for switching state–space models. Neural Comput. **12**, 831–864 (2000)
7. Goldberger, J., Roweis, S.: Hierarchical clustering of a mixture model. In: Advances in Neural Information Processing Systems (NIPS) (2005)
8. Hershey, J., Olsen, P.: Approximating the Kullback Leibler divergence between Gaussian mixture models. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2007)
9. Kaelbling, L., Littman, M., Cassandra, A.: Planning and acting in partially observable stochastic domains. Artif. Intell. **101**, 99–134 (1998)
10. Kullback, S.: A lower bound for discrimination in terms of variation. IEEE Trans. Inf. Theory **13**(1), 126–127 (1967)
11. Munos, R., Moore, A.: Variable resolution discretization for high-accuracy solutions of optimal control problems. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) (1999)
12. Oh, S., Rehg, J., Balch, T., Dellaert, F.: Data-driven mcmc for learning and inference in switching linear dynamic systems. In: Proceedings of the National Conference on Artificial Intelligence (AAAI) (2005)
13. Park, J., Sandberg, I.: Universal approximation using radial-basis-function networks. Neural Comput. **3**(2), 246–257 (1991)
14. Pineau, J., Gordan, G., Thrun, S.: Point-based value iteration: an anytime algorithm for POMPDs. In: International Joint Conference on Artificial Intelligence (IJCAI) (2003)
15. Porta, J., Spaan, M., Vlassis, N., Poupart, P.: Point-based value iteration for continuous POMDPs. J. Mach. Learn. Res. **7**, 2329–2367 (2006)
16. Shani, G., Brafman, R.I., Shimony, S.E.: Forward search value iteration for POMDPs. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) (2007)
17. Smith, T., Simmons, R.: Heuristic search value iteration for POMDPs. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI) (2004)
18. Smith, T., Simmons, R.: Point-based POMDP algorithms: improved analysis and implementation. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI) (2005)
19. Sondik, E.J.: The optimal control of partially observable Markov processes. Ph.D. Thesis, Stanford University (1971)
20. Spaan, M., Vlassis, N.: Perseus: Randomized point-based value iteration for POMDPs. J. Artif. Intell. Res. **24**, 195–220 (2005)
21. Thrun, S.: Monte carlo POMDPs. In: Advances in Neural Information Processing Systems (NIPS) (2000)
22. Zhang, K., Kwok, J.: Simplifying mixture models through function approximation. In: Advances in Neural Information Processing Systems (NIPS) (2006)
23. Zhang, N., Zhang, W.: Speeding up the convergence of value iteration in partially observable Markov decision processes. J. Artif. Intell. Res. **14**, 29–51 (2001)
24. Zhou, E., Fu, M.C., Marcus, S.I.: Solving continuous-state POMDPs vis density projection. IEEE Trans. Automat. Contr. **55**(5), 1101–1116 (2010)