2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Hyper-Particle Filtering for Stochastic Systems

James C. Davidson and Seth A. Hutchinson
{jcdavdsn, seth}@uiuc.edu
Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, Urbana, IL, USA

Abstract— **Information-feedback control schemes (more specifically, sensor-based control schemes) select an action at each stage based on the sensory data provided at that stage. Since it is impossible to know future sensor readings in advance, predicting the future behavior of a system becomes difficult. Hyper-particle filtering is a sequential computational scheme that enables probabilistic evaluation of future system performance in the face of this uncertainty. Rather than evaluating individual sample paths or relying on point estimates of state, hyper-particle filtering maintains at each stage an approximation of the full probability density function over the belief space (i.e., the space of possible posterior densities for the state estimate).**

By applying hyper-particle filtering, control policies can be more more accurately assessed and can be evaluated from one stage to the next. These aspects of hyper-particle filtering may prove to be useful when determining policies, not just when evaluating them.

I. INTRODUCTION

Whether predicting the effect of noise on the behavior of a robot or determining the effectiveness of a particular information-feedback control policy, a method is needed to predict the future behavior of partially observed stochastic systems. Most current methods either perform sample path simulation (as used in [1]–[8]), whereby a series of sample paths are generated and the ensemble average is taken to estimate the behavior, or the observation process is discarded entirely as in [9]–[11]. In such approaches, the behavior of the system is predicted without taking the effect of observations into consideration. Alternatively, the behavior is just predicted one stage into the future using forward projection techniques (refer to [12]). These techniques have various shortcomings when attempting to fully evaluate, more than one stage into the future, how the robot behaves from one future stage to the next. In such situations the observations can have a dramatic impact on the evolution on a robotic system. When considering information-feedback policies observations play a direct role in determining applied actions. Moreover, the observation itself molds the probability function over the state space. It is therefore critical that the full effect of the future observations be considered. Hyper-particle filtering has been developed to not only predict the behavior of a system but also to be used in the planning process.

Hyper-particle filtering is an approximation of the exact technique, which will be referred to as hyperfiltering. Hyperfiltering is a technique melding the concept of forward projection with the concept of filtering. Filtering propagates the behavior of the system and its uncertainty to the *current stage* for some known observations. Forward projection, on the other hand, propagates the predicted behavior of a system and its uncertainty forward from the current stage to the next *future stage*. Unfortunately, forward projection for partially observed stochastic systems for more than the next stage has received relatively little attention. This limitation is overcome by the formulation of the hyperfilter and the hyper-particle filter approximation method.

In this paper, insights from filtering are used to derive the concept of hyper-particle filtering. While filtering requires the retention of the conditional probability function at each stage, also known as the belief, hyperfiltering requires a construct capable of representing the additional level of uncertainty introduced when considering future unknown observations. The space of probability functions defined over the space of beliefs, henceforth the hyperbelief, is the precise construct needed: the hyperbelief can be propagated from one future stage to the next in a forward sequential manner.

While filtering is a difficult problem, the issues with hyperfiltering grow exponentially (quite literally in the case of discrete systems); the number of possible beliefs can grow exponentially with the time horizon. Another issue is the high nonlinearity of the belief transition probability. The hyper-particle filtering approximation method is introduced to mitigate these issues. Hyper-particle filtering retains a set of belief samples as well as a probability weight for each sample from one stage to the next. While hyper-particle filtering is based on particle filtering, hyper-particle filtering approximates the probability function over the belief space, whereas particle filtering approximates the evolution of a probability function over the state space.

Recently, researchers have sampled the belief space to obtain feasible/reachable beliefs to reduce the computational burden of finding nearly optimal policies (e.g., [2]–[5], [13]). These approaches sample a feasible set of the belief space by generating a random set of belief samples in the belief space. However, these methods do not retain probabilities associated with each sample and instead are are used to discretize the belief space into a meaningful finite set. They also fail to address the fundamental sequential nature that arises when a probability over the belief is evaluated forward over multiple stages. Like these methods, hyperfiltering focuses on on the possible beliefs as represented by the hyperbelief at each iteration. This is an important point and is one of the motivations for this research.

978-1-4244-1647-9/08/$25.00 ©2008 IEEE.

The hyper-particle filtering method will be introduced in Section IV. However, first background concepts are introduced in Section II followed by the formulation of hyperfiltering and an outline of potential applications in Section III. An illustrative example will be demonstrated in Section V. The document will then conclude with a comparison to other methods (in Section VI) and some final remarks (in Section VII).

II. Background: Model and Notation

Hyper-particle filtering performs sequential forward projection for partially observable Markov decision processes (POMDPs). POMDPs include at least the following components:

- The state space: $\mathcal{X}$.
- The finite set of control actions: $\mathcal{U}$.
- The transition probability function: $p_{\mathbf{x}_{k+1}|\mathbf{x}_k,u_k}$.
- The set of all possible observations: $\mathcal{Y}$.
- The observation probability function: $p_{\mathbf{y_k}|\mathbf{x}_k}$

at each stage k. In addition, a POMDP may be specified with a reward function $r(\cdot)$, which defines the objective to be optimized.

Ultimately, the goal of much of robotics research is to engineer autonomous or nearly autonomous systems. Within the context robotics, control theory, and AI this concept of autonomy comes to fruition via the motion strategy or control policy $\pi(\cdot)$. Hyper-particle filtering is developed to evaluate the effect of a policy on the performance of a system.

Algorithms for finding the exact optimal control policy for POMDPs have a best known computational time complexity that is exponential in the time horizon and the number of states [14]. Approximation techniques focus on reducing the computational complexity relating to one or both of the dimension of the belief space or the number of planes representing the value function (e.g., [1]–[8], [13], [15]–[21]).

These approximation techniques have at least one thing in common: the set possibilities that are evaluated is a subset of the complete set of possibilities that must be analyzed to find an exact solution. This is the case whether the set of possibilities evaluated is limited set of points in belief space to be considered or is a subset of the policies to be searched. In such situations, hyperfiltering may offer a tangible benefit in predicting the evolution of the system for approaches where local policies are used to plan between the set of possibilities.

III. Hyperfiltering

Hyperfiltering is a method, for systems modeled by POMDPs, to propagate the estimate of the belief and its uncertainty forward into future stages for unseen observations and unactualized control inputs. By choosing the probability function over the beliefs, hyperfiltering is able to sequentially evaluate the estimate of the system and its uncertainty forward from one stage to the next. Moreover, by adopting the complete representation of the uncertainty, instead of just some statistics of the belief, a more accurate representation of the evolution of the system is obtained.

The evolution of the probability function $p(x_k|I_k)$ at stage k, also known as the belief b_k (where $b_k \triangleq p_{\mathbf{x}_k|I_k}(\cdot, I_k)$) can be determined given the previous belief b_{k-1} and an applied control action $u_k \in \mathcal{U}$ via the belief transition function:

$$b_k = B(b_{k-1}, u_{k-1}, y_k),$$

where the belief transition function describes the Bayesian filtering over all states $x \in \mathcal{X}$, or

$$B(b_k, u_k, y_{k+1})(x_{k+1}) = \frac{p(y_{k+1}|x_{k+1}) \sum_{x_k \in \mathcal{X}} p(x_{k+1}|x_k, u_k) b_k(x_k)}{\sum_{x_{k+1} \in \mathcal{X}} p(y_{k+1}|x_{k+1}) \sum_{x_k \in \mathcal{X}} p(x_{k+1}|x_k, u_k) b_k(x_k)}. \quad (1)$$

The notation $B(\cdot)(x_{k+1})$ is adopted to represent the resulting function evaluated at a specific state x_{k+1}. The belief at each stage k resides in the belief space $\mathcal{P}_b$, which is the space of all possible beliefs. For discrete state POMDPs, the belief space is represented as an $|\mathcal{X}|-1$ dimensional simplex $\Delta^{|\mathcal{X}|-1}$, where $|\mathcal{X}|$ is the number of states in the state space.

When predicting future behavior, the observations are unknown and stochastic in nature. The future belief therefore becomes a random variable (refer to Def. III.1 below) defined by the stochastic process

$$\mathbf{b}_{k+1} = B(\mathbf{b}_k, u_k, \mathbf{y}_{k+1},). \quad (2)$$

The evolution from one stage to the next via the stochastic process (2) generates a random variable and, thus, a representation of the probability function over the belief is needed to proceed.

Definition III.1. *For a POMDP with a discrete state space and applied control policy π, the hyperbelief β_k at stage k is a functional, such that $\beta_k : \mathcal{P}_b \to \mathbb{R}^+$ and $\int_{b_k \in \mathcal{P}_b} \beta_k(b_k) db_k = 1$. The hyperbelief is a probability function over the belief space at each stage. The initial hyperbelief β_1 at stage $k = 1$ is given; for $k > 1$, the hyperbelief is defined as*

$$\beta_k \triangleq p_{\mathbf{b}_k|\beta_1,\pi}.$$

Each β_k is contained in the *hyperbelief space* $\mathcal{P}_\beta$. The hyperbelief space $\mathcal{P}_\beta$ is defined as the set of all probability measures over or $\mathbb{B}(\mathcal{P}_b)$, the Borel σ-algebra defined over the belief space $\mathcal{P}_b$.

For discrete state space POMDP systems, the state space, belief space, and hyperbelief space are all well defined. The belief space is represented as $\Delta^{|\mathcal{X}|-1}$. The Borel σ-algebra $\mathbb{B}(\Delta^{n-1})$ exists and, thus, the hyperbelief space is well defined.

The probabilistic outcome of the belief transition function, when predicting the behavior for future stages, is given by the belief transition probability function.

Definition III.2. *The belief transition probability function $p(b_{k+1}|b_k, u_k)$, represents the probability of the outcome b_{k+1} of the stochastic process $B(b_k, u_k, \mathbf{y}_{k+1})$ given b_k and*

the applied control input u_k, where $\mathbf{y}_k + 1$ is a random observation, which acts like a disturbance to the system.

Since both π and b_k are known, the probability function over the observations can be inferred. This induces a probability function on the set of actions, which is used to determine the probability of b_{k+1} occurring. Many of the approaches in the POMDP optimization literature either explicitly or implicitly use the belief transition probability equation (refer to [12]).

The hyperbelief β_{k+1} can be marginalized on the previous hyperbelief β_k to represent β_{k+1} as the integral of the belief transition probability function and the previous hyperbelief β_k at stage k. In turn, β_k can be represented as the integral of the belief transition probability function and the previous hyperbelief β_{k-1} at stage $k-1$. Thus, a sequential formulation of the hyperbelief can be obtained.

Defining Π to be the set of all information-feedback policies that depend on the state and $\mathcal{M}(\mathcal{P}_b)$ as the set of all bounded $\mathbb{B}(\mathcal{P}_b)$-measurable functions defined over $\mathcal{P}_b$, it is possible to establish a sequential formulation of the hyperbelief.

Theorem III.3. *For a system modeled as a POMDP with a discrete state space with a given control policy $\pi \in \Pi$, the hyperbelief $\beta_k \in \mathcal{P}_\beta$ at stage k given the initial hyperbelief $\beta_1 \in \mathcal{P}_\beta$ can be evaluated via the recursive application of the belief transition probability function from stage k to the initial stage. This holds if the belief transition function is defined such that $p_{b_{k+1}|b_k,u_k}(\cdot|b_k,u_k) \in \mathcal{P}_\beta$ for all $b_k \in \mathcal{P}_b$, $u_k \in \mathcal{U}$ and $p(\mathbf{b}_{k+1}|\cdot,u_k) \in \mathcal{M}(\mathcal{P}_b)$ for all $b_{k+1} \in \mathbb{B}(\mathcal{P}_b)$, $u_k \in \mathcal{U}$.*

The proof, omitted for brevity, follows by induction on the application of the belief transition function. Also, by elementary properties of integrable functions, the hyperbelief can be evaluated and is a probability function defined over the belief space.

Definition III.4. *The function that transfers a hyperbelief $\beta_k \in \mathcal{P}_\beta$ into the hyperbelief $\beta_{k+1} \in \mathcal{P}_\beta$ given a policy $\pi \in \Pi$ is denoted as the hyperbelief transition function Υ, such that $\Upsilon : \mathcal{P}_\beta \times \Pi \to \mathcal{P}_\beta$, where Π is the set of all information feedback policies. The hyperbelief transition function is represented as*

$$\beta_{k+1} = \Upsilon(\beta_k, \pi),$$

where, for each $b_{k+1} \in \mathcal{P}_b$,

$$\Upsilon(\beta_k,\pi)(b_{k+1}) \triangleq \int_{b_k \in \mathcal{P}_b} p\left(b_{k+1}|b_k,\pi\left(b_k\right)\right)\beta_k(b_k)db_k.$$

Because the output of the hyperbelief transition function is defined over the belief space $\mathcal{P}_b$, the notation $\Upsilon(\cdot)(b_{k+1})$ is adopted to represent the resulting function evaluated at a specific belief $b_{k+1} \in \mathcal{P}_b$.

The hyperbelief transition function can be nested as a set of compositions up to any given stage. In this way, it is possible to preserve the result of one stage as the input for the next stage. Hence, the hyperbelief encapsulates all the information needed to predict the evolution of a partially observed system to future stages.

A. Generating the belief transition probability function

The belief transition function is a composition of two steps: the prediction step and the update step, both of which are deterministic processes that transition a belief into another belief in the belief space. The prediction step updates the probability function based on an applied control action. The update step, on the other hand, reweights the probability function based on the observation.

The prediction step, represented by $\hat{b}_{k+1} = \hat{B}(b_k, u_k)$, transitions a belief b_k at stage k to a belief $\hat{b}_{k+1}$ at stage $k+1$ for some control action u_k. The prediction step, forall $\hat{x}_{k+1}$ in $\mathcal{X}$, is given as

$$\hat{B}(b_k,u_k)(\hat{x}_{k+1}) = \sum_{x_k \in \mathcal{X}} p(\hat{x}_{k+1}|x_k,u_k)b_k(x_k) \quad (3)$$

and, thus, $\hat{b}_{k+1} = p_{\mathbf{x}_{k+1}|I_k,u_k}$.

By substituting $\hat{b}_{k+1}$ into (1), the update step can be formulated. At stage $k+1$, the update step $\bar{B}$ transitions the belief $\hat{b}_{k+1}$ to belief b_{k+1} at stage $k+1$ for an observation y_{k+1}. The update step is given as

$$\bar{B}(\hat{b}_{k+1},y_{k+1})(x_{k+1}) = \frac{p(y_{k+1}|x_{k+1})\hat{b}_{k+1}(x_{k+1})}{\sum_{x_{k+1}\in\mathcal{X}} p(y_{k+1}|x_{k+1})\hat{b}_{k+1}(x_{k+1})}, \quad (4)$$

for all x_{k+1} in $\mathcal{X}$.

The composition of both (3) and (4) becomes the belief transition function:

$$B(b_k,u_k,y_{k+1}) = \bar{B}(\hat{B}(b_k,u_k),y_{k+1}).$$

In this way, the belief transition function can be split into two steps.

When the belief of the previous stage is random and the observation is random, the evolution of the system from one stage to the next still proceeds in two steps. The first step is the predicted transition of a random belief under a given policy. This is related to the prediction stage of traditional filtering methods. When conditioned on a specific belief, $\hat{B}$ condenses to a single point indicating the single, unique outcome. The probability of $\hat{b}_{k+1}$, therefore, is given as

$$p(\hat{b}_{k+1}|b_k,u_k) = \delta(\hat{b}_{k+1} - \hat{B}(b_k,u_k)). \quad (5)$$

Like the prediction step $\hat{B}$, when the update $\bar{B}$ is conditioned on a specific observation the probability function over the belief space condenses to a single point in the belief space indicating the single, unique outcome:

$$p(b_{k+1}|\hat{b}_{k+1},y_{k+1}) = \delta(b_{k+1} - \bar{B}(\hat{b}_{k+1},y_{k+1})).$$

However, the observation is unknown when predicting the future, so the observation $\mathbf{y}_{k+1}$ acts like a noise term in the update $\bar{B}(\hat{b}_{k+1},\mathbf{y}_{k+1})$. When taking the random observation into account, the resulting probability for some belief b_{k+1}

given some predicted belief $\hat{b}_{k+1}$ is given as

$$p(b_{k+1}|\hat{b}_{k+1}) = \sum_{y_{k+1}\in\mathcal{Y}} p(b_{k+1}|\hat{b}_{k+1}, y_{k+1})p(y_{k+1}|\hat{b}_{k+1}) \tag{6}$$

where y_{k+1} is introduced as a marginalizing term. The probability function $p(y_{k+1}|\hat{b}_{k+1})$ represents the posterior over the observation given the probability function $\hat{b}_{k+1}$ defined over the state space, which is evaluated as $p(y_{k+1}|\hat{b}_{k+1}) = \sum_{x_{k+1}\in\mathcal{X}} p(y_{k+1}|x_{k+1})\hat{b}(x_{k+1})$.

If the composition of both (5) and (6) is taken, it is possible to represent the belief transition probability function as

$$p(b_{k+1}|b_k, u_k) \tag{7}$$

$$= \int_{\hat{b}_{k+1}\in\mathcal{P}_b} p(b_{k+1}|\hat{b}_{k+1}, u_k)p(\hat{b}_{k+1}|b_k, u_k)d\hat{b}_{k+1} \tag{8}$$

$$= \int_{\hat{b}_{k+1}\in\mathcal{P}_b} p(b_{k+1}|\hat{b}_{k+1})\delta(\hat{b}_{k+1} - \hat{B}(b_k, u_k))d\hat{b}_{k+1} \tag{9}$$

$$= p(b_{k+1}|\hat{B}(b_k, u_k)). \tag{10}$$

By marginalizing the belief transition probability function on $\hat{b}_{k+1}$, (8) is obtained. The fact that the probability of b_{k+1} is conditionally independent of u_k given $\hat{b}_{k+1}$ is applied to derive (9) from (8). Next, (5) is substituted into the equation and, finally, the equation reduces to (10) because $\hat{b}_{k+1}$ is unique with probability one when conditioned on b_k. The integration reduces to the single point $\hat{b}_{k+1} \in \mathcal{P}_b$, which is obtained via $\hat{B}(b_k, u_k)$.

IV. Hyper-Particle Filtering

A. Hyper-particle filter formulation

Adapted from the concept of particle filtering (e.g., [22]–[33]), hyper-particle filtering approximates the hyperfiltering method. Hyper-particle filtering takes as input a set of hyper-particles (which approximate the hyperbelief) and a control policy and outputs a new set of hyper-particles via the hyperbelief transition probability function (as defined at Def. III.4).

The hyper-particle filter is a two tier approach. At the lower level, a traditional particle filter is used to approximate one possible belief over the state by a set of particles, generated using particle filtering. At the upper level, the hyperbelief is approximated by a set of hyper-particles. Each hyper-particle has both a sample and a weight associated to it. The sample is just an approximated belief represented by a particle set. The weight summed over the samples approximates probability of each sample. Besides being a two tier approach, the hyperfilter proceeds in two steps because sampling from belief transition probability function, $p(b_{k+1}|\ b_k, u_k)$, directly may not be practical or feasible. However, as shown in (10), $p(b_{k+1}\ |\ b_k, u_k) = p(b_{k+1}|\ \hat{B}(b_k, u_k))$, which can be used to generate samples in two steps: the prediction step and the update step.

At the upper level, the hyper-particle filter at stage k consists of a set of R hyper-particles. These hyper-particles are denoted as $\mathcal{Z}_k = \left\{z_k^i\right\}_{i=1}^R$. Each z_k^i is a pair where $z_k^i = (\alpha_k^i, b_k^i)$, with α_k^i a nonnegative scalar weight and $b_k^i \in \mathcal{P}_b$. Each b_k^i represents a point in belief space, and α_k^i represents a probability mass for that point. The set $\mathcal{Z}_k$ approximates the hyperbelief: $\beta_k(b_k) \approx p(b_k|\mathcal{Z}_k) = \sum_{i=1}^R \alpha_k^i \delta(b_k - b_k^i)$.

At the lower level, the belief point b_k^i is associated with a traditional particle set, where each b_k^i comprises a set of pairs of scalar weights w_k^q and samples x_k^q in the state space $\mathcal{X}$, giving: $b_k^i = \{w_k^q, x_k^q\}_{q=1}^Q$, where Q is the number of particle samples. Thus, each b_k^i approximates the belief at stage k as $b_k(x_k) \approx \sum_{q=1}^Q w_k^q \delta(x_k - x_k^q)$.

The evolution of the hyper-particle filtering proceeds as follows. At stage k, $\mathcal{Z}_k$ is an approximation of the hyperbelief β_k. The predicted next stage hyperbelief under the control policy π is approximated by using traditional particle filtering to approximate the predicted hyperbelief by sampling a set of hyper-particles from $\hat{B}$ for each hyper-particle in $\mathcal{Z}_k$.

The outcome is a new hyper-particle $\hat{z}_{k+1}^j$ generated for each $z_k^i \in \mathcal{Z}_k$, labeled $\hat{z}_{k+1}^j = \{\hat{\alpha}_{k+1}^j, \hat{b}_{k+1}^j\}$. Each $\hat{b}_{k+1}^j$ approximates the deterministic outcome of the prediction step $\hat{b}_{k+1}^j \approx \hat{B}(b_k^i, u_k)$. Each new hyper-particle is assigned a weight $\hat{\alpha}_{k+1}^j = \alpha_k^i$, where b_k^i is the belief that generated $\hat{b}_{k+1}^j$.

Once the predicted set of hyper-particles is generated, a set of at most T beliefs for each belief in $\hat{\mathcal{Z}}_{k+1}$ are randomly sampled to create a new set of hyper-particle samples: $\mathcal{Z}_{k+1} = \{\alpha_{k+1}^l, b_{k+1}^l\}_{l=1}^{RT}$ at stage $k+1$. These beliefs are sampled according to an importance sampling function $q_{b_{k+1}|\hat{b}_{k+1}}(\cdot|\hat{b}_{k+1}^j)$, for each j. The importance sampling function $q_{b_{k+1}|\hat{b}_{k+1}}$ is a random, or quasi-random, function that generates a sample in the belief space given $\hat{b}_{k+1}$. The importance sampling function can be biased based on a variety of desired attributes from emphasizing sampling of certain regions to forcing quasi-uniform sampling. For example, $p_{b_{k+1}|\hat{b}_{k+1}}$ can be used in the cases where it is possible to sample this probability function directly.

For each b_{k+1}^l, a new updated weight must be calculated, where the new weight is based on the previous weight and the probability of the new belief. For hyper-particle z_{k+1}^l the probability is is approximated by the weight α_{k+1}^l, which is found by

$$p(b_{k+1}^l|\beta_k, \pi) = \int_{b_k\in\mathcal{P}_b} p(b_{k+1}^l|b_k, \pi(b_k))p(b_k|\beta_k, \pi)db_k \tag{11}$$

$$= \int_{b_k\in\mathcal{P}_b} p(b_{k+1}^l|\hat{B}(b_k, \pi(b_k)))\beta_k(b_k)db_k \tag{12}$$

$$\approx \sum_{i=1}^R p(b_{k+1}^l|\hat{B}(b_k, \pi(b_k^i)))\alpha_k^i \tag{13}$$

$$\approx \sum_{j=1}^R p(b_{k+1}^l|\hat{b}_{k+1})\hat{\alpha}_{k+1}^j \tag{14}$$

$$\approx \eta_{k+1}\frac{p(b_{k+1}^l|\hat{b}_{k+1}^j)}{q(b_{k+1}^l|\hat{b}_{k+1}^j)}\hat{\alpha}_{k+1}^j = \alpha_{k+1}^l. \tag{15}$$

The first equation, (11), follows from marginalizing $p(b^l_{k+1}|\beta_k, \pi)$ on b_k, which reduces to $p(b^l_{k+1}|b_k, \pi)$. The fact that $p(b_{k+1}|b_k, u_k) = p(b_{k+1}|\hat{B}(b_k, u_k))$, as was shown in (10), was used to obtain (12). At each stage, the hyper-particle set $\mathcal{Z}_k$ is an approximation of the hyperbelief β_k. Thus, (12) can be approximated as the summation over all the belief samples in $\mathcal{Z}_k$ in (13). The set $\hat{\mathcal{Z}}_{k+1}$ was generated to approximate the output of $\hat{B}$ for each sample in $\mathcal{Z}_k$ and, thus, (14) is obtained by substituting $\hat{\mathcal{Z}}_{k+1}$ into (13), where $\alpha^i_k = \hat{\alpha}^j_{k+1}$. The normalizing constant η_{k+1} is given by $\frac{1}{\eta_{k+1}} = \sum_{l=1}^{RT} \alpha^l_{k+1}$.

Because each sample b^l_{k+1} was generated randomly, adverse effects are introduced by sampling from an importance sampling function instead of the transition and observation probability functions. The adverse effects result from a bias in the set of samples generated. Without taking into account the bias, the result can quickly become erroneous. By dividing $p_{\mathbf{b}_{k+1}|\hat{\mathbf{b}}_k}$ by $q_{\mathbf{b}_{k+1}|\hat{\mathbf{b}}_{k-1}}$, the bias is attenuated. As observed when performing Monte Carlo integration, for some function $c(\cdot)$,

$$E[c(b)] = \sum_{b \in \mathcal{P}_b} c(b)p(b) = \sum_{b \in \mathcal{P}_b} c(b)\frac{p(b)}{q(b)}q(b).$$

The expectation of a r.v. with a probability function $p(b)$ can be represented as the expectation of another r.v. with the probability function $q(b)$ by weighting $c(b)$ by the ratio of $p(b)$ and $q(b)$. This reduces or eliminates the the bias of the importance sampling on the expected value.

Additionally, (15) follows from the assumption that each particle b^l_{k+1} has zero probability of occurring from any belief sample other than the belief sample $\hat{b}^j_{k+1}$ that generated it, so the summation reduces to the evaluation over a single belief. This stage is analogous to the prediction stage performed by the particle filter.

The algorithm describing computation of one entire stage is described in Algorithm 1, where the algorithm for *HPF_sample* is given in Algorithm 2.

At this point, all that remains is to sample from the posterior probability over the belief space approximated by the hyper-particle set. This endows hyper-particle filtering with the ability to sample based on the likelihood of the outcome over the entire approximated hyperbelief, not just the prior probability over the belief space of a single sample (as is done in sample path simulation). Sampling from the approximated posterior performs two other functions: it reduces the hyper-particle degeneracy (refer to [30]) and limits the growth of the number of hyper-particles.

Sampling from the approximated posterior is a procedure wherein a new set of samples is generated from the current set of samples and is performed by generating the cumulative distribution function (cdf) over the hyper-particle set that is represented by $\mathcal{Z}_{k+1}$ at stage $k+1$. A starting point in the cdf is randomly generated in the range $s = \frac{1}{n}$, where n is the number of hyper-particles desired. Then for n samples, starting from s and adding $\frac{1}{n}$ for each sample, the belief with the probability nearest the probability of the sample is chosen. Finally a weight of $\frac{1}{n}$ is assigned to each new hyper-particle. This procedure is outlined in Algorithm 2.

Algorithm 1 Hyper-particle filter

$\bar{\mathcal{Z}}$ = HPF($\mathcal{Z}$,n, T,π), where $\mathcal{Z} = \{\alpha^i, b^i\}_{i=1}^{R}$
$l \leftarrow 1$
for $j = 1, \cdots, R$ **do**
  predict $\hat{b}^j$ using the particle filtering prediction
  $\hat{\alpha}^j \leftarrow \alpha^j$
  for $t = 1, \cdots, T$ **do**
    sample $\bar{b}^l$ from $q(\cdot|\ \hat{b}^j)$
    $l \leftarrow l + 1$
  end for
end for
for $l = 1, \ldots, RT$ **do**
  $\bar{\alpha}^l \leftarrow \frac{p(\bar{b}^l \ | \ \hat{b}^j)}{q(\bar{b}^l \ | \ \hat{b}^j)}\hat{\alpha}^j$
end for
$\alpha_{tot} \leftarrow \sum_{l-1}^{RT} \bar{\alpha}^l$
normalize each $\bar{\alpha}^l$ by α_{tot}
$\bar{\mathcal{Z}} \leftarrow \{\bar{\alpha}^l, b^l\}_{l=1}^{RT}$
$\bar{\mathcal{Z}} \leftarrow$HPF_sample($\bar{\mathcal{Z}}$,$n$)
return $\bar{\mathcal{Z}}$

Algorithm 2 Hyper-particle filter sample from posterior

$\mathcal{Z}$=HPF_sample($\bar{\mathcal{Z}}$,n), where $\bar{\mathcal{Z}} = \{\bar{\alpha}^i, \bar{b}^i\}_{i=1}^{v}$
Initialize cdf c to zero
for $j = 1, \cdots, v$ **do**
  $c_{j+1} \leftarrow c_j + \bar{\alpha}^j$
end for
draw sample u_1 from uniform density over $[0, \frac{1}{v}]$
$i \leftarrow 2$
for $j = 1, \cdots, n$ **do**
  $u_j \leftarrow u_j + \frac{j-1}{n}$
  while $u_j > c_i$ **do**
    $i \leftarrow i + 1$
  end while
  $a^j \leftarrow \frac{1}{n}$
  $b^j \leftarrow b^{i-1}$
end for
$\mathcal{Z} \leftarrow \{\alpha^j, b^j\}_{j=1}^{n}$
return $\mathcal{Z}$

B. Hyperfiltering algorithmic complexity

The computational complexity of the method varies, depending on the choice of performance parameters, from $O(KR(QL+M))$ to $O(R^K)$, where K is the time horizon, R is the number of hyper-particles, Q is the number of particles approximating the belief (via particle filtering), $O(L)$ is the computational time complexity of the particle filtering sampling, and $O(M)$ is the computational complexity of performing the hyper-particle sampling. In Algorithm 1, T is the number of intermediate observation samples. If n remains constant, then the number of hyperparticles is fixed and each stage has $O(R(QL + M))$ complexity. However, if $n = RT$ at each stage, then there is in the worst case an

exponential increase in the number of particles. If $p_{x_{k+1}|x_k,u_k}$ is chosen as the particle filtering importance sampling function and $p_{b_{k+1}|\hat{b}_{k+1}}$ is chosen as the hyper-particle filtering importance sampling function, the entire process then has a $O(K|\mathcal{X}|RQ|\mathcal{Y}|)$ computational complexity, as generating $p_{b_{k+1}|\hat{b}_{k+1}}$ takes $O(|\mathcal{X}|Q|Y|)$ time. In this case, it is better generate a belief sample for each $y_{k+1} \in \mathcal{Y}$ as it has the same complexity but represents the exact transition.

V. Results

Hyper-particle filtering is first performed to benchmark the performance of hyper-particle filtering on a set of problems that are prevalent in the POMDP literature in Section V-A. As a future avenue of this research is to use hyper-particle filtering to aid in finding nearly optimal event-driven policies, the performance of the hyper-particle filter for a system subject to a sensor-based event-driven control policy is explored in Section V-B.

A. Evaluation for various typical POMDP problems

An exhaustive evaluation of the performance of the hyper-particle filtering method was performed for three representative problems. The problems vary in size:

- 4×4 grid: *comprising 16* states, 2 observations, and 4 actions
- *Hallway2*: comprising 92 states, 17 observations, and 5 actions
- *Tag*: comprising 870 states, 30 observations, and 5 actions

These problems were chosen to demonstrate how the hyper-particle filtering method performs as the number of states and observations vary.

To obtain a meaningful analysis of the quality of the hyper-particle filtering technique, the performance of each example was evaluated relative to the optimal or nearly-optimal policy for the objective function associated with each problem. For the $4x4$ example, the optimal solution was found using the Incremental Pruning method described in [34] using pomdp-solve [35]. Unfortunately, the other examples (i.e., *Hallway2* and *Tag*) are too large to find exact solutions, so approximate solutions were found using zmdp [36], which implements the HSVI algorithm of [4].

The performance is evaluated by finding the statistical average of the expected total reward and standard deviation of the expected total reward taken over series of simulations (30 iterations for 4×4 and *Hallway2* and 10 iterations for *Tag*). For the hyper-particle filtering method, the expected total reward is obtained by evaluating the stochastic expected value of the reward over the hyperbelief at each stage.

For each of the examples, a particle filtering approximation of the belief is employed and simulations are performed to compare the effect of particle filtering on the precision of result. This allows not only the effect of just the variation in the number of hyper-particles to be fully evaluated but also the effect of the variation in the number of particles. As can be seen in Figure 1, the reward and variance quickly converge as the number of hyper-particles increase. The effect of the number of particles seems to be influential as well, but to a lesser degree. It is also interesting to note that the quality of the approximation stabilizes for just a few particles and hyper-particles. This may imply that only a small number of particles and hyper-particles are needed to obtain reasonable results for problems with relatively few observations and states.

B. Evaluation for an exemplar event-driven example

To evaluate the hyper-particle for a more realistic problem,a representative, complex system is analyzed using the hyper-particle filter technique. This example comprises 10,000 states, eight actions, and 328,420 observations. A substantial amount of uncertainty exists in both the transition probability function and in the observation probability function. Such a problem is immense by the standards of the problems in the literature (e.g., [1]–[6], [13], [15]–[20]).

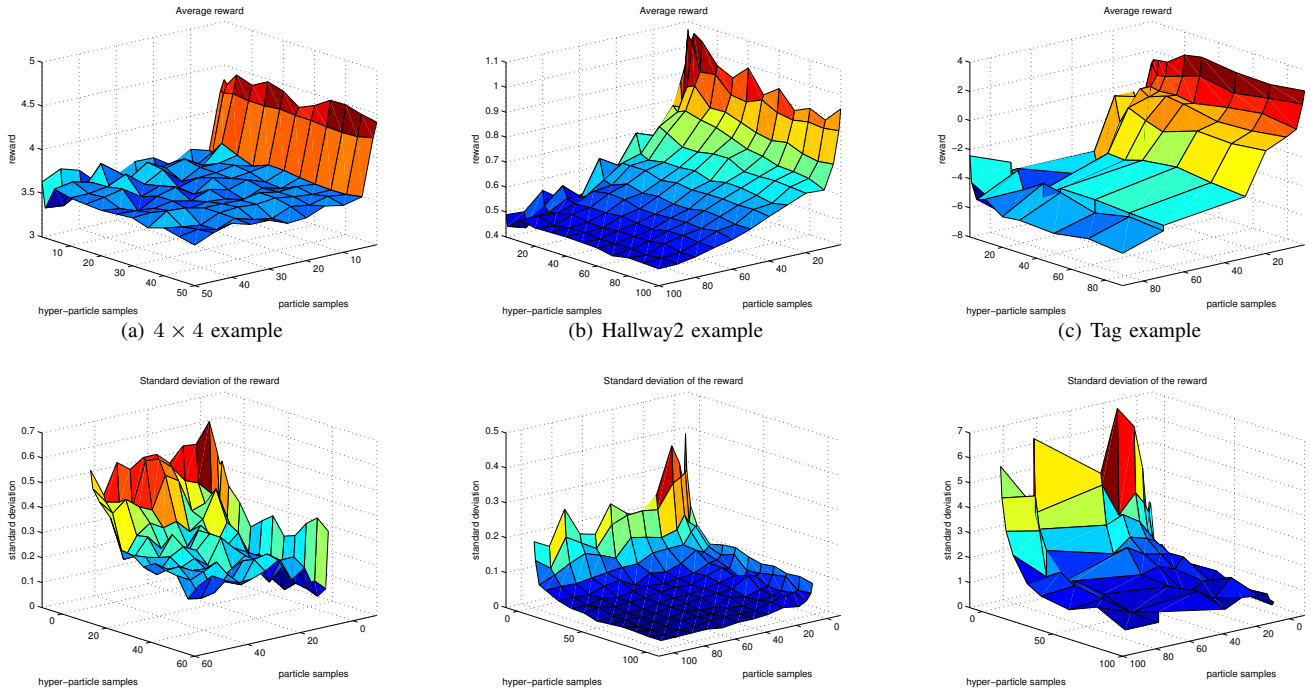
In this example, the continuous state space (a subset of $\mathbb{R}^2$) is approximated by a discrete 100×100 grid. The policy employed for this simple example comprises an event-based controller, where the events are based in the observation space.

To understand to what extent the accuracy of the predicted performance is affected, an examination of this example was performed in order to determine the effect on the performance as the number of particles and hyper-particles samples vary. The convex sum of the entropy (as described in [37]) and squared L2-norm from expected position to the goal was chosen as the objective function because it incorporates both a measure of the uncertainty and a measure of the distance between the robot and the goal state. The results illustrated in Figure 2(a) depict the average reward over the set of iterations as well as the standard deviation in the reward for a varying number of hyper-particles, with each hyper-particle represented by a particle set with 100 particles. In Figure 2(b), the performance is evaluated for a fixed number of 50 hyper-particles and a variable number of particles representing each hyper-particle.

Interestingly, the cost, as can be seen in Figure 2(b), increases as the number of particles increases. This result is logical considering the representation. When there are fewer particles, the probability function is represented by a small set of points, artificially making the probability function appear more condensed and, hence, more certain. However, as the number of samples increases, the representation of the probability function becomes more accurate and the uncertainty present in the system becomes better represented, which is reflected in the expected reward.

Because a significant amount of variation exists in the problem, as illustrated at Figure 2(a), the standard deviation is rather large and only appears to converge for more than 60 hyper-particles. However, as demonstrated above for the earlier examples in Section sec:comparison-examples, the hyper-particle filtering approach converges quickly as the number of hyper-particles increases.

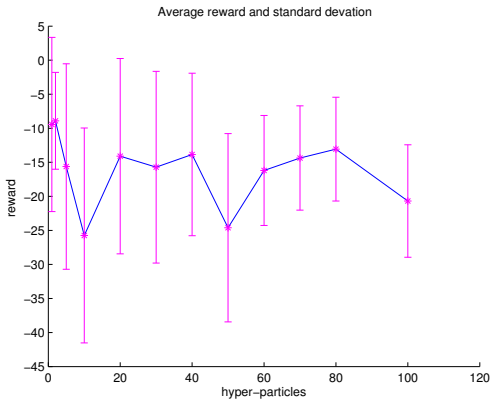
This event-driven example was chosen not only to demonstrate the effectiveness of the hyper-particle filtering method



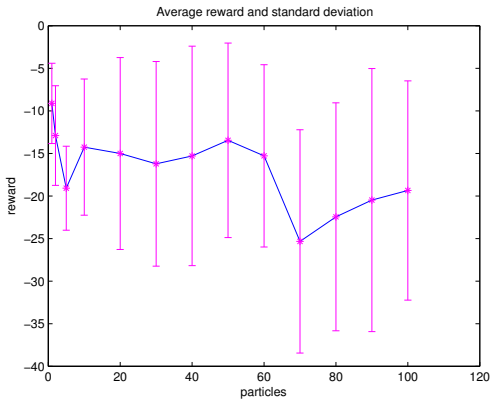
(a) 4×4 example (b) Hallway2 example (c) Tag example

(d) 4×4 example (e) Hallway2 example (f) Tag example

Fig. 1. Average reward and standard deviation



(a)



(b)

Fig. 2. Results for a varying number of hyper-particles

but to assess its applicability in evaluating event-driven policies. While in this example the policy is given, future research will use hyperfiltering to aid in automatically determining nearly optimal policies using event-driven policies, like the one used in this example.

VI. Discussion

Hyper-particle filtering is more than just a sequential implementation of sample path simulation. Hyper-particle filtering samples from the posterior over the beliefs generated from the approximated hyperbelief. Sample path simulation, on the other hand, sample a single observation from each prior belief. By sampling from the posterior over the belief, a more accurate representation of the next stages hyperbelief can be obtained. While in some respects, sample path simulation and hyper-particle filtering appear similar, they are fundamentally different approaches. Hyper-particle filtering is an approach to approximating the probability function over the belief at each stage, whereas sample path simulation generates a series of sample paths to a given stage. Still, there are some algorithmic similarities: both generate random samples and both are forward-based methods. By treating the sample set as an approximation over the belief space at the current stage, the resampling algorithm of the hyper-particle filter reduces degeneracy and focuses the sampled set on the more likely future beliefs. Sample path methods lack this feature entirely and any unlikely sample paths are never eliminated. Moreover, when analyzing the performance for a given objective function, the reward at each stage is taken as the expectation over the hyper-particle set, unlike sample

path methods, where the reward for each simulation is the reward of the path.

VII. Conclusions

A method for approximating the hyperfiltering algorithm was presented. Hyper-particle filtering is a method for predicting the evolution of a stochastic partially observed Markov decision processes forward into future stages. This is achieved via a sequential method, whereby probability function over the belief is propagated forward one future stage to the next for a given policy.

By representing the probability function over the beliefs at each stage, hyperfiltering can be employed to analyze the effect that observations have on the evolution of a system in addition to answering queries regarding how sensing capabilities affect the evolution or discerning which sensing configurations are sufficient to obtain a desired objective.

Besides the potential application of determining the utility of information, the hyperfiltering technique may be useful in finding nearly optimal solutions for POMDP systems. There are various methods to approximate POMDPs to find nearly optimal solutions by reducing the set of possibilities explored including: considering only a subset of the points in belief or information space that need to be searched and by considering only a subset of the possible policies. Hyper-particle filtering may be used in in these scenarios as a tool to propagate the uncertainty from one future stage to the next. Thus, hyper-particle filtering can be used to evaluate the connectivity from one belief point to the next and it can be used to evaluate a finite set of possible policies.

References

[1] Z. Feng and E. Hansen, "Approximate planning for factored POMDPs," in *Proceedings of the Sixth European Conference on Planning (ECP-01)*, 2001, pp. 409–416.

[2] D. Bernstein, E. Hansen, and S. Zilberstein, "Bounded policy iteration for decentralized POMDPs," in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 52–57.

[3] S. Thrun, "Monte Carlo POMDPs," in *Advances in Neural Information Processing Systems*, 2000, pp. 1064–1070.

[4] T. Smith and R. Simmons, "Heuristic search value iteration for POMDPs," in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, 2004, pp. 520–527.

[5] T. Smith and R. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2005.

[6] N. Roy and G. Gordon, "Exponential family PCA for belief compression in POMDPs," in *Advances in Neural Information Processing Systems*, 2002, pp. 1–8.

[7] M. T. Spaan and N. Vlassis, "A point-based POMDP algorithm for robot planning," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 2399–2404.

[8] M. T. Spaan and N. Vlassis, "PERSEUS: Randomized point-based value iteration for POMDPs," in *Journal of Artificial Intelligence Research*, vol. 24, 2005, pp. 195–220.

[9] M. Erdmann and M. Mason, "An exploration of sensorless manipulation," *IEEE Journal of Robotics and Automation*, vol. 4, no. 4, pp. 369–379, 1988.

[10] M. Morari and J. Lee, "Model predictive control: Past, present, and future," *Computers and Chemical Engineering*, vol. 23, pp. 667–682, 1997.

[11] G. Shani, R. I. Brafman, and S. E. Shimony, "Forward search value iteration for POMDPs," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2007.

[12] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: The MIT Press, 2005.

[13] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*, 2003, pp. 1025 – 1032.

[14] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, Jan. 1998.

[15] M. L. Littman, "Memoryless policies: Theoretical limitations and practical results," in *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, 1994, pp. 238–247.

[16] D. A. McAllester and S. Singh, "Approximate planning for factored POMDPs using belief state simplification," in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999, pp. 409–416.

[17] P. Poupart and C. Boutilier, "Value directed compression of POMDPs," in *Advances in Neural Information Processing Systems*, 2003.

[18] E. A. Hansen, "An improved policy iteration algorithm for partially observable MDPs," in *Advances in Neural Information Processing Systems*, vol. 10, 1998, pp. 1015–1021.

[19] E. Hansen, "Solving POMDPs by searching in policy space," in *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, 1998, pp. 211–21.

[20] N. Meuleau, L. Peshkin, K.-E. Kim, and L. P. Kaelbling, "Learning finite-state controllers for partially observable environments," in *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, 1999, pp. 427–436.

[21] P. Poupart and C. Boutilier, "Bounded finite state controllers," in *In Advances in Neural Information Processing Systems (NIPS)*, 2003.

[22] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. New York, NY: Springer Verlag, 2001.

[23] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 50, no. 2, pp. 174–188, February 2002.

[24] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, pp. 107–113, 1993.

[25] C. Kwok, D. Fox, and M. Meila, "Real-time particle filters," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 469–484, March 2004.

[26] S. Thrun, "Particle filters in robotics," in *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence*, 2002.

[27] D. Fox, "Adapting the sample size in particle filters through kld-sampling," *International Journal of Robotics Research (IJRR)*, vol. 22, pp. 985–1003, 2003.

[28] J. H. Kotecha and P. M. Djuric, "Gaussian sum particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2602–2612, Oct. 2003.

[29] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[30] A. Doucet, "On sequential simulation-based methods for Bayesian filtering," Department of Engineering, University of Cambridge, Tech. Rep. CUED/F-INFENG, TR. 310, 1998.

[31] D. Crisan, P. D. Moral, and T. Lyons, "Discrete filtering using branching and interacting particle systems," *Markov Processes and Related Fields*, vol. 5, no. 3, pp. 293–318, 1999.

[32] K. Kanazawa, D. Koller, and S. Russell, "Stochastic simulation algorithms for dynamic probabilistic networks," in *Proceedings of the Conference on Uncertainty in Artifical Intelligence*, 1995, pp. 346–351.

[33] D. Crisan and A. Doucet, "Convergence of sequential Monte Carlo methods," Cambridge University, Tech. Rep. CUED/FINFENG, TR381, 2000.

[34] A. Cassandra, M. L. Littman, and N. L. Zhang, "Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes," in *Proceedings of Uncertainty in Artificial Intelligence*, 1997, pp. 54–61.

[35] T. Cassandra, "POMDP solver software: pomdp-solve v. 5.3," 2007. [Online]. Available: http://pomdp.org/pomdp/code/index.shtml

[36] T. Smith, "ZMDP software for POMDP and MDP planning: ZMDPv. 1.1.3," 2007. [Online]. Available: http://www.cs.cmu.edu/ trey/zmdp/

[37] R. B. Ash, *Information Theory*. New York, NY: Dover Publications, 1990.