Continuous Time Particle Filtering

Brenda Ng
Harvard University
Cambridge, MA 02138
bmng@eecs.harvard.edu

Avi Pfeffer
Harvard University
Cambridge, MA 02138
avi@eecs.harvard.edu

Richard Dearden
University of Birmingham
Birmingham B15 2TT, UK
R.W.Dearden@cs.bham.ac.uk

Abstract

We present the continuous-time particle filter (CTPF) – an extension of the discrete-time particle filter for monitoring continuous-time dynamic systems. Our methods apply to hybrid systems containing both discrete and continuous variables. The dynamics of the discrete state system are governed by a Markov jump process. Observations of the discrete process are intermittent and irregular. Whenever the discrete process is observed, CTPF samples a trajectory of the underlying Markov jump process. This trajectory is then used to estimate the continuous variables using the system dynamics determined by the discrete state in the trajectory. We use the unscented Kalman-Bucy filter to handle nonlinearities and continuous time. We present results showing that CTPF is more stable in its performance than discrete-time particle filtering, even when the discrete-time algorithm is allowed to update many more times than CTPF. We also present a method for online learning of the Markov jump process model that governs the discrete states.

1 Introduction

The work described here is directly motivated by problems encountered in creating a state estimation system for the K-9 experimental Mars rover at NASA Ames Research Center [Willeke and Dearden, 2004]. As with many robotic systems, K-9 contains numerous sensors that report on its performance. These sensors produce telemetry at different rates, and frequently at rates that vary over time. As a result, using a fixed time-step state estimator, for example a Kalman filter, is extremely difficult. In this paper we describe a continuous-time state estimation algorithm based on particle filtering that can handle telemetry arriving at different rates and changes in system behavior between observations. The system can also update its model of system behavior to improve its performance and to detect gradual degradation of the system.

State estimation is typically done either using discrete diagnosis algorithms such as Livingstone [Williams and Nayak, 1996], or more recently, using hybrid system representations. The discrete approach is well suited for domains where there are occasional transitions between relatively stable states, and where sensor readings tend to be very reliable. Unfortunately, this is not the case for K-9 and discrete approaches have largely proven unhelpful [Washington *et al.*, 2000]. For this reason, we use a hybrid system model for state representation and particle filtering for state estimation.

Probabilistic hybrid automata (PHAs) [Hofbaur and Williams, 2002] are commonly used for representing hybrid systems in diagnosis and state estimation (other representations have similar expressive power). In the PHA model, a system consists of a set of discrete *modes* and a set of continuous variables. For each mode, a set of differential equations describes the continuous behavior in that mode. Transitions between modes are stochastic; a transition matrix represents the probability of being in state s' at time $t+1$ given that the system is in state s at time t. Transition probabilities may depend on the values of the continuous variables.

The problem with applying the PHA model (and other similar representations) to systems such as K-9 is the assumption that a transition matrix applied at fixed time intervals is sufficient to describe the evolution of the system. This implies that only one discrete transition can occur per interval, and also that it doesn't matter when in the interval the transition occurs. These assumptions lead to two problems. First, they make the system model unwieldy as they force a single transition matrix to represent the effects of multiple discrete transitions. Second, they reduce efficiency by forcing model steps to be small enough for the assumptions to be reasonable. The continuous time model avoids these assumptions, allowing state updates to be performed only when a new observation arrives or when a significant event occurs.

The approach we take is to build a continuous-time model of the system (see Section 2). As before, there is a differential equation model governing the continuous system behavior for every discrete system mode, but now the discrete state transitions are represented as a continuous-time Markov jump process. That is, we keep a set of transition probabilities for each discrete mode, as well as the distribution over *how long* the system will remain in the mode before transitioning out.

As we said above, we use particle filtering to track the system state (see Section 4). For efficiency, and to make it possible to track large systems, we use Rao-Blackwellized particle filters [Doucet *et al.*, 2000a], in which the discrete mode is sampled, just as in a standard particle filter, but the continuous state is tracked using an unscented Kalman-Bucy filter.

This work addresses the key challenge that many mode changes on the rover, as with most systems, are not observable directly. Commanded changes such as moving from

stopped to driving are observable, but the occurrence of faults, and other mode transitions triggered by the environment or the continuous state, do not produce telemetry directly, and can only be inferred from their effects on the continuous system behavior. The algorithm we have developed to address these difficulties, the *continuous-time particle filter*, is described in Section 5.

One major advantage of the continuous-time approach is that it allows resource-bounded computation. Although the cost of a single update may be higher than that of a discrete update (because of the Kalman-Bucy filter equations), computation need only be performed when new observations arrive. Indeed, if computation is limited, observations can even be ignored until sufficient computation is available to update the estimate. For discrete approaches, an update must be performed for each time step, even if no telemetry has arrived.

In Section 6, we discuss how the parameters of the continuous-time Markov jump process can be learned by observing the system. Section 7 applies the approach to the simulated rover model presented in Section 3.

2 Continuous-time hybrid-state processes

A continuous-time hybrid-state process consists of interacting discrete-state and continuous-state random variables.

The discrete-state variables $\mathbf{Z}_t$ evolve according to dynamics described by continuous-time Markov jump processes. A Markov jump process is a random variable Z_t that is parametrized by time $t \in [0, \infty)$. Z_t starts in an initial state z and remains in this state for a random amount of time before it makes a transition to a different state. The time that Z_t stays in a particular state is exponentially distributed, due to the Markovian property of the process.

Mathematically, a Markov jump process Z_t is characterized by its *intensity matrix*

$$Q = \begin{bmatrix} -q_1 & q_{12} & \dots \\ q_{12} & -q_2 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

where

$$q_{ij} = \lim_{\Delta t \to 0} \frac{P\{Z_{t+\Delta t} = j | Z_t = i\}}{\Delta t}, \ i \neq j \text{ and } q_i = \sum_{j \neq i} q_{ij}$$

in which q_{ij} is the transition rate that defines the probability per time unit that the system makes a transition from state i to state j and q_i is the total transition rate out of state i. The amount of time that the process stays in state i is distributed according to the exponential distribution $f_i(t) = q_i \exp(-q_i t)$. When the process leaves state i, it enters the next state j with probability $P_{ij} = \frac{q_{ij}}{\sum_j q_{ij}}$ if $i \neq j$ and 0 otherwise. In this paper, we assume that the process is stationary, i.e. the intensity matrix is the same at all times.

In recent work, [Nodelman *et al.*, 2002] introduced continuous-time Bayesian networks (CTBNs) as a way to factorize Markov jump processes over discrete-valued variables. Our work extends the inference to hybrid-state processes.

The instantiation of $\mathbf{Z}_t$ corresponds to a unique set of equations that govern the dynamics of the continuous variables $\mathbf{X}_t$. Let $\tau_1, \tau_2, \dots$ denote the times at which $\mathbf{Z}_t$ changes

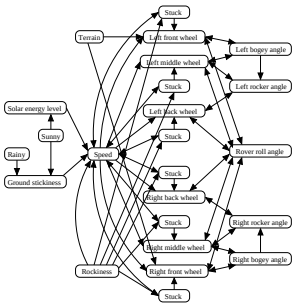

Figure 1: Rover CTBN. Note that a CTBN may contain cycles.

value. Let $\mathbf{z}_i$ denote the value of $\mathbf{Z}_t$ in the interval $[\tau_i, \tau_{i+1}]$. During the time span of $[\tau_i, \tau_{i+1}]$ the variables $\mathbf{X}_t$ follow nonlinear dynamics given by

$$\frac{d}{dt}\mathbf{X}_t = F_{\mathbf{z}_i}(\mathbf{X}_t, \mathbf{U}_t) + W_t \quad (1)$$

and the observations $\mathbf{Y}_t$ are given by

$$\mathbf{Y}_t = H_{\mathbf{z}_i}(\mathbf{X}_t, \mathbf{U}_t) + V_t \quad (2)$$

where $W_t \sim \mathcal{N}(0, \Phi_t)$ and $V_t \sim \mathcal{N}(0, \Psi_t)$ are the respective process and measurement noise, and $\mathbf{U}_t$ is the control input.

In this work, we assume that some variables in $\mathbf{Z}_t$ may be observed, but only at random discrete points in time. On the other hand, $\mathbf{Y}_t$ are observed more frequently, but as noisy measurements of the true continuous state $\mathbf{X}_t$.

3 A continuous-time rover model

As a testbed, we use a continuous-time rover model. The rover employs a rocker-bogey suspension system for its six wheels. On each side, the front two wheels are attached to the bogey, which pivots around its center. The center of the bogey is attached to the rocker, which is attached to the main rover chassis and the rear wheels. Motion from the bogey can cause the rocker to pivot around the differential axle, a mechanism that centers the rover chassis between the left and right sets of wheels. From the rocker angles and the bogey angles, we can infer the rover orientation and thus the wheel heights relative to the rover chassis.

We are interested in reasoning about the rover wheel dynamics under the effects of weather, terrain, ground conditions and faulty wheel behavior. The model consists of 20 state variables and 4 transient variables. There are two groups of state variables: 5 that model background conditions and 15 that directly model the rover. The background variables model environmental factors such as weather, terrain, ground rockiness and ground stickiness, while the rover variables model the solar energy level available to the rover, the rover speed, the wheel heights, the wheel stuck conditions, and the rover roll angle. The CTBN structure is shown in Figure 1.

The model comprises 11 binary and ternary discrete variables and 9 continuous variables. The only observable discrete variables are $Sunny$, $Rainy$, $Terrain$ and $Rockiness$. The observation of these variables at sparse time points corresponds to situations when the rover takes measurements of its surroundings, to plan navigation or to manage power consumption. The wheel stuck states are never observed and are

inferred from observations of the continuous wheel variables. The wheel stuck condition is more frequently induced when the rover travels at high speed and when the ground is rocky.

The rover dynamics are adapted from the kinematic analysis in [Hacot, 1998]. The front wheels receive an input proportional to some height perturbation induced by the terrain. The change in the middle wheel height is proportional to the difference between the front and middle wheel height, taken relative to the rover body which experiences roll (torsional rotation) on uneven terrain. To illustrate the system's nonlinearity, the equation for the middle wheel height is:

$$\frac{d}{dt}z_t^M = \kappa_t \cos(\Theta_t)\left(l_1 \sin(\theta_t^B) - l_2 \sin(\theta_t^B + \beta)\right) \quad (3)$$

where Θ_t is the roll angle, θ_t^B is the bogey angle, and l_1, l_2 and β are constant parameters of the physical rover model. The change in the back wheel is defined in a similar manner. For each wheel, the proportionality constant κ_t is dependent on the speed and whether the wheel is stuck. The speed is affected by the availability of solar energy and by ground surface characteristics such as stickiness and rockiness.

Noisy measurements of the continuous variables that model the bogey and rocker angles are observed more frequently than the discrete observations. The bogey angle is determined from the front and middle wheel heights. The rocker angle depends on the middle and back wheel heights as well as the bogey angle.

4 Preliminaries

Let the hybrid system state be represented as $\mathbf{S}_t = \{\mathbf{Z}_t, \mathbf{X}_t\}$. We denote $y_{1:t}$ as the history of observations from time 1 to time t. The aim is to track the probability $p(\mathbf{S}_t|y_{1:t})$ that any given state may be the true system state, conditioned upon the sequence of past observations $y_{1:t}$. This probability distribution, also referred to as the *belief state*, can be recursively estimated given the previous belief state $p(\mathbf{S}_{t-1}|y_{1:t-1})$ and the current observation y_t:

$$p(\mathbf{S}_t|y_{1:t}) = p(y_t|\mathbf{S}_t)\frac{p(\mathbf{S}_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} \quad (4)$$

where $p(\mathbf{S}_t|y_{1:t-1}) = \int p(\mathbf{S}_t|\mathbf{S}_{t-1})p(\mathbf{S}_{t-1}|y_{1:t-1})d\mathbf{S}_{t-1}$. However, the computation of this integral leads to intractability in all but the smallest conditional linear Gaussian models. As a result, one must resort to approximate inference methods to compute the updated belief state.

4.1 Particle filtering

Particle filtering (PF) [Doucet *et al.*, 2000b] approximates (4) by a discrete sum of *particles* or samples of possible states drawn from that distribution:

$$p(\mathbf{S}_t|y_{1:t}) \approx \sum_{i=1}^{N} \frac{1}{N}\delta(s_t^{(i)}) \quad (5)$$

where $\delta(\cdot)$ denotes the Dirac delta function. Since it is difficult to sample from $p(\mathbf{S}_t|y_{1:t})$ directly, importance sampling is used, in which particles are drawn from a more tractable *proposal* distribution, then each particle is weighted to account for this bias. Since variance increases as the state space increases, the number of particles required to achieve decent accuracy increases as well, thus making PF an expensive solution for tracking complex, high-dimensional systems.

Rao-Blackwellization [Doucet *et al.*, 2000a] is a technique that improves PF by analytically marginalizing out some of the variables. This method is especially applicable to our domain since the structure of the rover model can be factorized:

$$p(\mathbf{S}_t|y_{1:t-1}) = \int p(\mathbf{X}_t|\mathbf{Z}_t, \mathbf{X}_{t-1})p(\mathbf{Z}_t|\mathbf{S}_{t-1})p(\mathbf{S}_{t-1}|y_{1:t-1})d\mathbf{S}_{t-1} \quad (6)$$

Hence, *Rao-Blackwellized Particle Filtering* (RBPF) can be used to sample only from the (lower-dimensional) discrete distribution and the continuous distribution can subsequently be computed using the Kalman filter.

4.2 The unscented Kalman-Bucy filter

The Kalman filter [Grewal and Andrews, 2001] is an efficient, recursive method that finds the least-square estimate of the state of a discrete-time, linear stochastic process with Gaussian noise. The Kalman-Bucy filter is the continuous-time counterpart to the discrete-time Kalman filter. It uses a differential equation to generate an estimate $\hat{\mathbf{x}}_t$ of the continuous state, and generates from that an estimate $\hat{\mathbf{y}}_t$ of the observation. It also generates P_t, the covariance of the state estimate error $\mathbf{x}_t - \hat{\mathbf{x}}_t$, and R_t, the covariance of $\mathbf{y}_t - \hat{\mathbf{y}}_t$. The update of P_t also uses a differential equation. Given these estimates, the probability of the observation $\mathbf{y}_t$ is given by a normal distribution with mean $\hat{y}_t$ and covariance R_t.

The Kalman-Bucy filter assumes a linear transition model and observation model. To handle nonlinear models, we adopt an approach similar to the discrete-time unscented Kalman filter (UKF) [Wan and van der Merwe, 2000] and extend the Kalman-Bucy filter by applying the unscented transformation [Julier and Uhlman, 2002] to the state estimation. Thus, the *unscented Kalman-Bucy filter* (UKB) is a continuous-time filter that allows for nonlinear models. UKB is used in RBPF as an approximation to the Kalman-Bucy filter. This approximation introduces bias and does not provide the theoretical guarantee that variance is reduced in RBPF.

5 The continuous-time particle filter

In the continuous-time particle filter (CTPF), we perform an update each time a discrete observation is received. If no discrete-state process is observed, updates can be triggered by an asynchronous request for update. Technically, this is treated the same way as observing the discrete-state process with a vacuous observation. If the new observation is received at time t_n and the previous observation was at time t_{n-1}, an update means generating particles of the state at t_n, using the particles generated at t_{n-1}. We assume that observations of the discrete-state processes are sparse and intermittent. We make no assumption that observations come at regular intervals or that the observations coincide with any discrete transitions. Let ζ_n be the observation at time t_n. We may observe ζ_n as the entirety of $\mathbf{Z}_{t_n}$, or ζ_n may be an assignment of values to a subset of the variables in $\mathbf{Z}_{t_n}$, or ζ_n may be vacuous, assigning values to none of the variables. In any case, we will

say that a value $\mathbf{z}$ agrees with the observation ζ_n if it assigns the same value to all variables assigned by ζ_n.

At a high level, the CTPF works as follows. When updating from t_{n-1} to t_n, we sample a trajectory from the Markov jump process for the time interval $[t_{n-1}, t_n]$. This results in a set of time points $\tau_0 = t_{n-1}, \tau_1, \ldots, \tau_k = t_n$ at which $\mathbf{Z}$ changes value. Let $\mathbf{z}_i$ denote the value of $\mathbf{Z}$ in interval $[\tau_i, \tau_{i+1}]$. We then perform Rao-Blackwellization to obtain updates for the continuous variables. We go through the sequence of intervals, starting at $[\tau_0, \tau_1]$ and ending at $[\tau_{k-1}, \tau_k]$, propagating our estimate of the mean and variance of the continuous variables $\mathbf{X}$ using UKB. In the time period $[\tau_i, \tau_{i+1}]$, the continuous dynamics are governed by the value $\mathbf{z}_i$. As we propagate the continuous variables, we also compute the probability of the trajectory, which is used as the particle weight. We repeat this process to obtain a set of weighted particles at time t_n, and resample from these to get a new set of unweighted particles.

The behavior of the Markov jump process governing $\mathbf{Z}$ is characterized by the intensity matrix $Q_{\mathbf{Z}}$. The process of generating a trajectory of the Markov jump process from time t_{n-1} to t_n is as follows. For each particle at time t_{n-1}, let $\mathbf{z}_0$ be the value of $\mathbf{Z}$ in that particle. Set $\tau_0 = t_{n-1}$ and $i = 0$. We use the intensity matrix to sample a time from the exponential distribution $f_{\mathbf{z}_i}(t) = q_{\mathbf{z}_i} \exp(-q_{\mathbf{z}_i} t)$ to obtain the length of time γ until $\mathbf{Z}$ next changes value. If $\tau_i + \gamma < t_n$, we sample $\mathbf{z}_{i+1}$ from the intensity matrix, and continue to loop, setting $\tau_{i+1} = \tau_i + \gamma$. Otherwise, we know that $\mathbf{Z} = \mathbf{z}_i$ at t_n. If this value agrees with the evidence ζ_n, the particle is kept, otherwise it is rejected.

Once we have the jump process, we proceed to propagate the continuous variables $\mathbf{X}$ through the time points $\tau_0, \ldots, \tau_k$. Since there may be many observations of the continuous variables in the interval $[\tau_i, \tau_{i+1}]$, when we propagate them from τ_i to τ_{i+1}, we need to consider all the observations between those times. Let δ_0 be τ_i, δ_ℓ be τ_{i+1} and $\delta_1, \ldots, \delta_{\ell-1}$ be the time points at which continuous observations were received. We loop through the intervals $[\delta_j, \delta_{j+1}]$ in turn. For each interval, we propagate the mean and variance of the continuous variables forward using UKB. We also obtain the probability of the observation at time δ_{j+1}, given by the predictive density $p(\mathbf{y}_{\delta_{j+1}} | \mathbf{y}_{\delta_1:\delta_j}) = \mathcal{N}(\mathbf{y}_{\delta_{j+1}}; \hat{\mathbf{y}}_{\delta_{j+1}}, R_{\delta_{j+1}})$ where $\hat{\mathbf{y}}_{\delta_{j+1}}$ and $R_{\delta_{j+1}}$ are the estimate and covariance of $\mathbf{y}_{\delta_{j+1}}$ under UKB. We then multiply the probability of all the observations together over all time points δ_j in $[\tau_i, \tau_{i+1}]$ and over all τ_i to obtain the probability of the sequence of observations. This becomes the weight of the particle.

We repeat this procedure for each of the M particles from time t_{n-1}. Each particle consists of a candidate Markov jump process for each discrete variable in $\mathbf{Z}_t$, and the mean and covariance for the continuous variables in $\mathbf{X}_t$. The Markov jump process candidates that are far from the ground truth will lead to poor tracking of the continuous variables due to incorrect parametrization of the continuous-state dynamics. Particles associated with the ill-fit Markov jump processes will have low weights and will be resampled out.

The pseudocode for CTPF is shown in Figure 2.

let M be the number of particles
let N be the number of discrete observations
for n = 1 **to** N **do**
  Dynamics propagation
  for m = 1 **to** M **do**
    let ζ_n be the n^{th} observation
    let t_n be the observed time of ζ_n
    Simulate a Markov jump process
    let $i = 0, \tau_0 = t_{n-1}$
    let $\mathbf{z}_0$ be the value of $\mathbf{Z}$ in the m^{th} particle at time t_{n-1}
    while $\tau_i < t_n$ **do**
      Sample γ from the PDF $f(t) = q_{\mathbf{z}_i} \exp(-q_{\mathbf{z}_i} t)$
      if $\tau_i + \gamma < t_n$ **then**
        let $\tau_{i+1} = \tau_i + \gamma$
        Sample $\mathbf{z}_{i+1}$ using the intensity matrix $Q_{\mathbf{Z}}$
      else
        let $\tau_{i+1} = t_n$; **let** $\mathbf{z}^{(m)} = \mathbf{z}_i$
      $i \leftarrow i + 1$
    if $\mathbf{z}^{(m)}$ disagrees with ζ_n, reject the particle
    let $k = i$
    Propagate continuous variables
    let $\hat{\mathbf{x}}_{t_{n-1}}$ and $P_{t_{n-1}}$ be the mean and covariance of the continuous variables in the m^{th} particle at time t_{n-1}
    let $w^{(m)} = 1$
    for i = 0 **to** $k - 1$ **do**
      let $\delta_1, \ldots, \delta_{\ell-1}$ be the times of continuous observations in $[\tau_i, \tau_{i+1}]$
      let $\delta_0 = \tau_i$, $\delta_\ell = \tau_{i+1}$
      for j = 0 **to** $\ell - 1$ **do**
        Compute $\hat{\mathbf{x}}_{\delta_{j+1}}$, $P_{\delta_{j+1}}$, $\hat{\mathbf{y}}_{\delta_{j+1}}$ and $R_{\delta_{j+1}}$ using UKB beginning with $\hat{\mathbf{x}}_{\delta_j}$ and P_{δ_j}, under the dynamics determined by $\mathbf{z}_i$
        $w^{(m)} \leftarrow w^{(m)} \times \mathcal{N}(\mathbf{y}_{\delta_{j+1}}; \hat{\mathbf{y}}_{\delta_{j+1}}, R_{\delta_{j+1}})$
    let $\hat{\mathbf{x}}^{(m)}$ and $P^{(m)}$ be $\hat{\mathbf{x}}_{t_n}$ and P_{t_n}
  Resampling
  Normalize $w^{(m)} \leftarrow \frac{w^{(m)}}{\sum w^{(m)}}$
  Select M new particles where the probability of $(\mathbf{z}^{(m)}, \hat{\mathbf{x}}^{(m)}, P^{(m)})$ is $w^{(m)}$

Figure 2: Continuous-time particle filter

6 Learning

In real-world applications, we may not know the exact parametrization of $\mathbf{Z}_t$. Instead, we may have a prior distribution over the intensity matrices $Q_{\mathbf{Z}}$ and need to fine tune our estimate of $Q_{\mathbf{Z}}$ as observations about $\mathbf{Z}_t$ are made.

In this case, we model $Q_{\mathbf{Z}}$ as a random variable and we attribute a probability distribution $\phi(Q_{\mathbf{Z}})$ over the space of admissible intensity matrices, $\mathcal{Q}_Z$. Since the observations are exponentially distributed, we model $\phi(Q_{\mathbf{Z}})$ as a Gamma distribution since Gamma and exponential are conjugates.

Assume that the elements of $Q_{\mathbf{Z}}$ are distributed according to $q_{ij} \sim \Gamma\left(\frac{1}{\beta_i}, \alpha_{ij}\right)$. The prior distribution $Q_{\mathbf{Z}}$ is given by:

$$\phi(Q_{\mathbf{Z}}) \propto \prod_{i=1}^{m} \prod_{j \neq i} q_{ij}^{\alpha_{ij}-1} e^{-q_{ij}\beta_i} \tag{7}$$

Upon the arrival of a discrete-state observation ζ_n, we simulate a Markov jump process J over the time interval $[t_{n-1}, t_n]$. From J, we can compute the likelihood of $Q_{\mathbf{Z}}$:

$$L(Q_{\mathbf{Z}}) = \prod_{i=1}^{m} \prod_{j \neq i} q_{ij}^{N_{ij}} e^{-q_{ij} R_i} \tag{8}$$

where N_{ij} is the number of transitions from state i to j in the process J and R_i is the time spent in state i during process J. From this, we update the $Q_{\mathbf{Z}}$-distribution

$$\psi(Q_{\mathbf{Z}}) = L(Q)\phi(Q) = \prod_{i=1}^{m} \prod_{j \neq i} q_{ij}^{[N_{ij}+\alpha_{ij}]-1} e^{-q_{ij}[R_i+\beta_i]} \tag{9}$$

Upon the arrival of the next discrete-state observation, we sample $q_{ij} \sim \Gamma\left(\frac{1}{R_i+\beta_i}, N_{ij}+\alpha_{ij}\right)$ to get an instantiation for $Q_{\mathbf{Z}}$ that reflects the updated distribution.

The learning procedure is integrated into algorithm 2 as follows: we sample $Q_{\mathbf{Z}} \sim \phi(Q_{\mathbf{Z}})$ right before line (8) and perform the $Q_{\mathbf{Z}}$-distribution update, as prescribed in Equations (7)-(9), after line (20) of the pseudocode.

This procedure may seem like magic. After all, we are using statistics generated from simulating from $Q_{\mathbf{Z}}$ itself to obtain an updated estimate of $Q_{\mathbf{Z}}$. The reason it works is that not all $Q_{\mathbf{Z}}$s from the previous set of particles have the same chance of being accepted into the next set of particles. First of all, if the trajectory disagrees with the discrete observation ζ_n, it will be rejected. Second, the trajectories will be weighted by the probability of the continuous observations, and these weighted particles will be resampled. A trajectory, and thus a new intensity matrix, that agrees well with the observations, is more likely to be kept. In this way, the set of intensity matrices that are kept will be influenced by the observations, both discrete and continuous.

Other learning approaches have been explored in [Nodelman *et al.*, 2003] for parameter learning of CTBNs, and in [Bladt and Sørensen, 2003] for statistical inference of discretely observed Markov jump processes. However, these approaches are offline learning approaches where the sequence of discrete-time observations are assumed known a priori to the learning process. Our approach integrates online learning into the particle filtering framework, where we also take advantage of the continuous observations in eliminating poor candidates for $Q_{\mathbf{Z}}$. This advantage allows for relatively quick convergence to the neighborhood of the true $Q_{\mathbf{Z}}$ value.

7 Experimental results

We evaluate the performance of the CTPF algorithm on a simulated small model and the rover model. The small model consists of one discrete ternary variable, two continuous state variables, and two continuous observation variables. Depending on the state of the discrete-valued process, the continuous behavior can be either linear, polynomial or sinusoidal. The low dimensionality of the small model makes it feasible to compare against the discrete-time particle filter and to evaluate the performance of the learning procedure.

Figure 3 shows an experiment comparing CTPF to the discrete-time particle filter (DTPF). The results shown are averaged over 50 runs. The first graph shows the performance of CTPF with 10 particles and 55 updates, comparing the estimate of one of the continuous state variables produced by CTPF with the actual value of that variable. While CTPF updates the discrete variables at infrequent intervals, DTPF performs an update at equally-spaced intervals fixed by a preset time granularity. Both algorithms perform continuous inference, via UKB or UKF, at many intermediate time points, corresponding to when the continuous variables are observed.

The plotted points are the estimates at these intermediate points. From the graph, we see that CTPF is consistently able to track the variable through the different mode changes. The second graph (DTPF1) shows DTPF's performance using the same number of particles and the same number of updates as CTPF. We see that DTPF1 has worse tracking throughout, and does particularly poorly from time 75 to time 100. CTPF outperforms DTPF because it is able to perform updates precisely when they are needed.

DTPF is considerably faster than CTPF because it does not require solving differential equations. This was exacerbated by a slow Matlab implementation of the Kalman-Bucy filter, compared to a well-optimized implementation of the Kalman filter. We expect this difference to lessen with a more efficient implementation of CTPF. Nonetheless, we ran experiments that allowed DTPF the same running time as CTPF.

We first allowed DTPF to use more particles than CTPF, while maintaining the same number of updates. The results of DTPF, with 58 particles and 55 updates, are shown as DTPF2, in the third graph of Figure 3. When the number of updates is not sufficient for DTPF to track the variables, increasing the number of particles does not particularly help. We then allowed DTPF to update more frequently than CTPF, but use the same number of particles. The results, with 10 particles and 584 updates, are shown in the fourth graph and are labeled DTPF3. The DTPF3 results show a significant improvement in the performance of DTPF, although DTPF still does not perform as well as CTPF when the continuous variables are oscillating rapidly. Furthermore, we found the variance of the estimates in DTPF3 to be higher than that produced by CTPF.

Figure 4 shows the results of learning the Q matrix on the small model. The results were quite promising. The first graph compares the estimation error achieved by CTPF with learning the Q matrix and CTPF with using the true Q matrix. Both sets of CTPF experiments used 10 particles. We see that the error of CTPF with learning comes close to that of CTPF without learning after 1000 time steps. The second graph shows the distance of the learned Q matrix from the true Q matrix over time, computed as the sum of the element-wise differences between the two matrices. The graph shows the learned Q matrix converging towards the true Q matrix.

Lastly, we tested the CTPF algorithm on our main application, the continuous-time rover model. Despite the nonlinearity of the model, CTPF is able to track the wheel heights relatively well with 10 particles. The results of the CTPF inference, averaged over 20 runs, are presented in Figure 5 (one graph per rover wheel). The tracking error is mainly due to incorrect estimation of the wheel stuck variable. Whenever

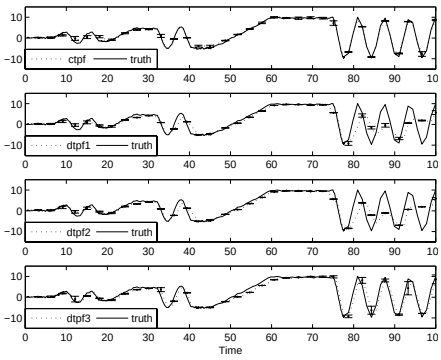

Figure 3: Estimation results

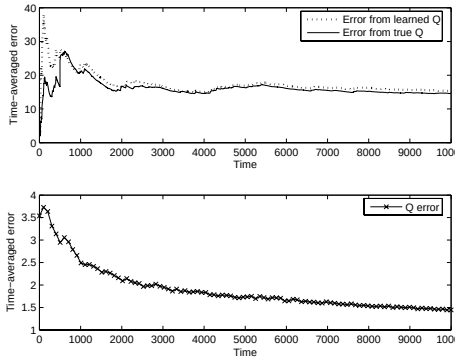

Figure 4: Learning results

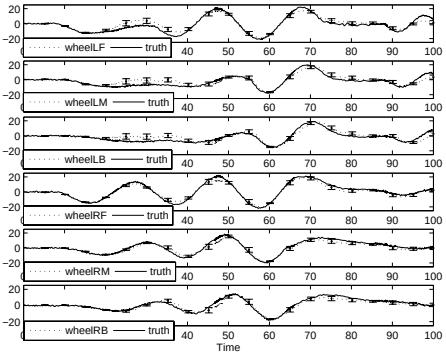

Figure 5: CTPF results on the rover

the CTPF estimate drifts from the ground truth, it is able to quickly recover from the error and continue to closely track the wheel heights. The empirical results confirm CTPF's applicability for monitoring complex continuous-time systems.

8 Conclusion and future work

Since most real-world systems evolve in continuous time, CTPF is a natural way to address the time granularity problem that is often associated with systems that have components that evolve at different rates. By performing state estimation in continuous time, CTPF is also better suited for resource-bounded computation since inference is no longer performed at every fixed time step (as in the discrete-time particle filter). Our results show that CTPF can effectively track the state of complex systems such as the rover. Moreover, when comparison is possible with discrete-time approaches, CTPF tracks the state more accurately. We believe that the approach is an important step in making recent advances in particle filtering applicable to a much larger set of continuous-time problems.

An important aspect in hybrid system modelling is the issue of *autonomous transitions*, in which the continuous state triggers a discrete transition. We model this in our approach by making the intensity matrices dependent on the continuous variables. This leads to difficulties in that while the system is in a state, the distribution over the time the system remains there may vary. At present we approximate this by only computing the intensity matrix at updates or when the discrete state changes. This is consistent with our aim of resource-bounded computation. We plan to investigate other alternatives such as resampling from an updated intensity matrix when the current distribution deviates drastically from the assumed distribution used for sampling.

Other areas of future work include applying this approach with factored state spaces [Ng *et al.*, 2002] to reduce the complexity of the Kalman-Bucy updates, and implementating this approach onboard an actual robot.

References

[Bladt and Sørensen, 2003] M. Bladt and M. Sørensen. Statistical inference for discretely observed markov processes. Technical report, University of Copenhagen, 2003.

[Doucet *et al.*, 2000a] A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proc. of the 16th UAI*, 2000.

[Doucet *et al.*, 2000b] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 2000.

[Grewal and Andrews, 2001] M. S. Grewal and A. P. Andrews. *Kalman Filtering: Theory and Practice Using MATLAB*. Wiley-Interscience, 2001.

[Hacot, 1998] H. Hacot. Analysis and traction control of a rocker-bogie planetary rover. Master's thesis, MIT, 1998.

[Hofbaur and Williams, 2002] M. W. Hofbaur and B. C. Williams. Hybrid diagnosis with unknown behaviour modes. In *Proc. of the 13th Intl. Workshop on Principles of Diagnosis*, 2002.

[Julier and Uhlman, 2002] S. Julier and J. Uhlman. The scaled unscented transformation. In *Proc. of the IEEE American Control Conference*, 2002.

[Ng *et al.*, 2002] B. Ng, L. Peshkin, and A. Pfeffer. Factored particles for scalable monitoring. In *Proc. of the 18th UAI*, 2002.

[Nodelman *et al.*, 2002] U. Nodelman, C. R. Shelton, and D. Koller. Continuous time bayesian networks. In *Proc. of the 18th UAI*, 2002.

[Nodelman *et al.*, 2003] U. Nodelman, C. R. Shelton, and D. Koller. Learning continuous time Bayesian networks. In *Proc. of the 19th UAI*, 2003.

[Wan and van der Merwe, 2000] E. Wan and R. van der Merwe. The unscented kalman filter for nonlinear estimation. In *Proc. of IEEE Symposium*, 2000.

[Washington *et al.*, 2000] R. Washington, K. Golden, and J. L. Bresina. Plan execution, monitoring, and adaptation for planetary rovers. *Electron. Trans. Artif. Intell.*, 2000.

[Willeke and Dearden, 2004] T. Willeke and R. Dearden. Building hybrid rover models: Lessons learned. In *Proc. of the 15th Intl. Workshop on Principles of Diagnosis*, 2004.

[Williams and Nayak, 1996] B. C. Williams and P. P. Nayak. A model-based approach to reactive self-configuring systems. In *Proc. of the 13th AAAI and the 8th IAAI*, 1996.