Real-time Fault Detection and Situational Awareness for Rovers: Report on the Mars Technology Program Task

Richard Dearden, Thomas Willeke
{RIACS, QSS Group Inc.} / NASA Ames Research Center
M.S. 269-3, Moffett Field, CA 94035, USA
{dearden,twilleke}@email.arc.nasa.gov

Frank Hutter
Fachbereich Informatik
Technische Universität Darmstadt, Germany
mail@fhutter.de

Reid Simmons, Vandi Verma
Carnegie Mellon University
Pittsburg, PA 15213, USA
{reids,vandi}@cs.cmu.edu

Sebastian Thrun
Computer Science Department, Stanford University
353 Serra Mall, Gates Bldg 154
Stanford, CA 94305, USA
thrun@stanford.edu

Abstract— An increased level of autonomy is critical for meeting many of the goals of advanced planetary rover missions such as NASA's 2009 Mars Science Lab. One important component of this is state estimation, and in particular fault detection on-board the rover. In this paper we describe the results of a project funded by the Mars Technology Program at NASA, aimed at developing algorithms to meet this requirement. We describe a number of particle filtering-based algorithms for state estimation which we have demonstrated successfully on diagnosis problems including the K-9 rover at NASA Ames Research Center and the Hyperion rover at CMU. Because of the close interaction between a rover and its environment, traditional discrete approaches to diagnosis are impractical for this domain. Therefore we model rover subsystems as hybrid discrete/continuous systems. There are three major challenges to make particle filters work in this domain. The first is that fault states typically have a very low probability of occurring, so there is a risk that no samples will enter fault states. The second issue is coping with the high-dimensional continuous state spaces of the hybrid system models, and the third is the severely constrained computational power available on the rover. This means that very few samples can be used if we wish to track the system state in real time. We describe a number of approaches to rover diagnosis specifically designed to address these challenges.

TABLE OF CONTENTS

1 INTRODUCTION
2 HYBRID DIAGNOSIS USING PARTICLE FILTERS
3 RISK-SENSITIVE PARTICLE FILTERS
4 VARIABLE RESOLUTION PARTICLE FILTER
5 RAO-BLACKWELLIZED PARTICLE FILTERS
6 FUTURE WORK

0-7803-8155-6/04/$17.00/©2004 IEEE
IEEEAC paper # 1209

1. INTRODUCTION

This paper reports the results from a study done as part of NASA's Mars Technology Program entitled *"Real-time Fault Detection and Situational Awareness for Rovers"*. The main goal of this project was to develop and demonstrate a fault-detection technology capable of operating on-board a Mars rover in real-time.

Fault diagnosis is a critical task for autonomous operation of systems such as spacecraft and planetary rovers. The diagnosis problem is to determine the state of a system over time given a stream of observations of that system. A common approach to this problem is *model-based diagnosis* [3], [4], in which the overall system state is represented as an assignment of a *mode* (a discrete state) to each component of the system. Such an assignment is a possible description of the current state of the system if the set of models associated with the modes is consistent with the observed sensor values. An example model-based diagnosis system is Livingstone [24], which flew on the Deep Space One spacecraft as part of the Remote Agent Experiment [15] in May 1999. In Livingstone, diagnosis is done by maintaining a candidate hypotheses (in other systems more than one hypothesis is kept) about the current state of each system component, and comparing the candidate's predicted behaviour with the system sensors. Traditional approaches operate on discrete models and use *monitors* to translate continuous sensor readings into discrete values. The monitors are typically only used once the sensor readings have settled on a consistent value, and hence these systems cannot generally diagnose transient events.

For many applications, e.g. planetary rovers, the complex dynamics of the system make reasoning with a discrete model inadequate. This is because too fine a discretization is required to accurately model the system; because the monitors would need global sensor information to discretize a single sensor correctly; and because transient events must be diagnosed. To overcome this we need to reason directly with the continuous values we receive from sensors: Our model needs to be a hybrid system.

A hybrid system consists of a set of discrete *modes*, which represent fault states or operational modes of the system, and a set of continuous variables which model the continuous quantities that affect system behaviour. We will use the term *state* to refer to the combination of these, that is, a state is a mode plus a value for each continuous variable, while the *mode* of a system refers only to the discrete part of the state. For example, consider a motor. It can be idle or powered, and has a number of fault modes such as having a faulty encoder. These correspond to the discrete part of the model. It also has continuous state, such as its running speed, the current powering it, and so on. In each discrete mode, there is a set of differential equations that describe the relationship between the various continuous values, and the way those values evolve over time. There is also a transition function that describes how the system moves from one mode to another. In many cases, not all of the hybrid system will be observable. Therefore, we also have an observation function that defines the likelihood of an observation given the mode and the values of the continuous variables. All these processes are inherently noisy, and the representation reflects this by explicitly including noise in the continuous values, and stochastic transitions between system modes. We describe our hybrid model in more detail below.

The complex dynamics of the rover, along with its interaction with an extremely complex, poorly modeled, and noisy environment—the surface of Mars—makes it very difficult to determine the true state of the rover at any point in time with certainty. To combat this, we advocate diagnosis algorithms that explicitly represent uncertainty at every point, thus allowing control of the rover to reason about the uncertainty when selecting actions to perform. This is extremely important for the rover as actions that appear good for the most likely state may be catastrophic if the rover turns out to be in another of its possible states. Explicitly representing uncertainty about state also makes diagnosis easier as we automatically have a set of alternate states when a new observation is inconsistent with the most likely state or states.

To represent uncertainty about the state of the rover, the diagnosis algorithms we will present maintain a belief distribution—a probability distribution over the states the rover could be in. To maintain this distribution, the algorithms will perform *Bayesian belief updating*. In this approach, we begin with a prior probability distribution $P(S_0) = \Phi$ that represents our initial beliefs about the state of the system, and as a sequence of observations $y_{1:t}$ are made of the system, we update the distribution to produce $P(S_t|\Phi, y_{1:t})$, the probability at time t of each state given the prior and the observations. Unfortunately, as we will see below, doing this computation exactly is computationally infeasible on-board the rover, so we must approximate it.

We will approximate Bayesian belief updating using a *particle filter* [10], [6]. A particle filter approximates the belief distribution using a set of point samples. In contrast, the popular Kalman Filter (see for example [7]) approximates the distribution by a single Gaussian distribution. The particle filter has a number of important advantages:

- It can be applied more easily to hybrid models. The particle filter simply maintains a discrete and continuous state for every sample, so the whole is a distribution over the complete model. Banks of Kalman filters—one for each discrete state—can also be used, but there is no simple way to determine the contribution of each filter to the overall distribution.
- It can represent non-Gaussian distributions. This allows models with non-linear continuous dynamics and non-Gaussian noise. As we shall see, these are important considerations for the rover domain.
- It can easily be adjusted to available computation, simply by increasing or decreasing the number of samples. This can be done on the fly as the algorithm is running.

The essence of the particle filter approach is to simulate the behaviour of the system. Each sample predicts a future behaviour of the system in a Monte-Carlo fashion, and the samples that match the observed system behaviour are kept, while ones that fail to predict the observations tend to die out. We describe the basic particle filter algorithm in Section 2.

The new algorithms we have developed for this project are motivated by a number of problems with applying the standard particle filter to diagnosis problems:

1. **Very low prior fault probabilities:** Diagnosis problems are particularly difficult for approximation algorithms based on sampling because the low probabilities of transitions to fault states can lead to incorrect diagnoses because there are no samples in a state even though it has a non-zero probability of occurring.
2. **Restricted computational resources:** For space applications, computation time is often at a premium, particularly for on-board real-time diagnosis. For this reason, diagnosis must be as efficient as possible.
3. **High dimensional state spaces:** As the dimensionality of a problem grows, the number of samples required to accurately approximate the posterior distribution grows exponentially.
4. **Non-linear stochastic transitions and observations:** Many algorithms are restricted to linear models with Gaussian noise. Our domains frequently behave non-linearly, so we would prefer an algorithm without this restriction.
5. **Multimodal system behaviour:** Even in a single discrete mode, the observations are often consistent with several values for the continuous variables, and so multi-modal distributions appear. For example, when a rover is commanded to accelerate, we are often uncertain about exactly when the command is executed. Different start times lead to different estimates of current speed, and hence a multi-modal distribution.

While these last two points are not a problem for the standard particle filter, many of the more efficient particle filter vari-

ants rely on assumptions inconsistent with them. Since non-linear dynamics and multimodal behaviour often occur in our domains, we would like to take advantage of the efficiency of these approaches without their representational restrictions.

In this report we present three algorithms each designed to address some of these problems. The three algorithms are all somewhat complementary, in the sense that ideas from all three can be combined into a single system. In Section 3 we present the risk-sensitive particle filter, an algorithm motivated by Problem 1. Section 4 looks at Problems 2 and 3, applying an approach based on abstraction in which system states are aggregated together in a hierarchy, and the full complexity of the individual state models is only looked at in detail if there is sufficient evidence that the rover is actually in that state. The third algorithm we present is motivated by Problems 1 and 2, and is based on the recently developed Rao-Blackwellized particle filter. However, that algorithm is restricted to linear-Gaussian models. We present the Gaussian particle filter, which removes this restriction, thus tackling Problems 4 and 5 as well. We conclude in Section 6 and describe planned future work on putting all these techniques together, and ways to make more progress on Problem 3, the least-well addressed by our current algorithms.

Hybrid Systems Modeling

Following [8] and [13], we model the system to be diagnosed as a discrete-time probabilistic hybrid automaton (PHA):

- $Z = z_1, \ldots, z_n$ is the set of discrete modes the system can be in.
- $X = x_1, \ldots, x_m$ is the set of continuous state variables which capture the dynamic evolution of the automaton. We write $P(Z_0, X_0)$ for the prior distribution over Z and X.
- Y is the set of observable variables. We write $P(Y_t|z_t, x_t)$ for the distribution of observations in state (z_t, x_t).
- There is a transition function that specifies:

$$P(Z_t|z_{t-1}, x_{t-1})$$

the conditional probability distribution over modes at time t given that the system is in state (z, x) at $t-1$. In some systems, this is independent of the continuous variables:

$$P(Z_t|z_{t-1}, x_{t-1}) = P(Z_t|z_{t-1})$$

- We write $P(X_t|z_{t-1}, x_{t-1})$ for the distribution over X at time t given that the system is in state (z, x) at $t-1$.

We denote a hybrid state of the system by $s = (z, x)$, which consists of a discrete mode z, and an assignment to the state variables x.

Diagnosis of a hybrid system of this kind is determining, at each time-step, the *belief state* $P(S_t|y_{1:t})$, a distribution that, for each state s, gives the probability that s is the true state of the system, given the observations so far. In principle, belief state tracking is an easy task, which can be performed using the *forward pass* equation:

$$\begin{aligned} P(s_t|y_{1:t}) &= \alpha P(y_t|s_t) \int P(s_t|s_{t-1}) P(s_{t-1}|y_{1:t-1}) ds_{t-1} \\ &= \alpha P(y_t|z_t, x_t) \\ &\int P(x_t|z_t, x_{t-1}) P(z_t|z_{t-1}, x_{t-1}) P(s_{t-1}|y_{1:t-1}) ds_{t-1} \end{aligned}$$

where α is a normalizing constant. Unfortunately, computing the integral exactly is intractable in all but the smallest of problems, or in certain special cases. The most important special case is a unimodal linear model with Gaussian noise. This is solved optimally and efficiently by the Kalman filter (KF). We describe the KF below; then, we weaken the model restrictions and describe algorithms for more general models, such as Particle Filters and Rao-Blackwellized Particle Filters. We end with the most general problem for which we propose the Gaussian Particle Filter.

2. HYBRID DIAGNOSIS USING PARTICLE FILTERS

When the system we want to diagnose has only one discrete mode, linear transition and observation functions for the continuous parameters and Gaussian noise there exists a closed form solution to the tracking problem. In this case, the belief state is a multivariate Gaussian and can be computed incrementally using a *Kalman filter* (KF). At each time-step t the Kalman filtering algorithm updates sufficient statistics $(\mu_{t-1}, \Sigma_{t-1})$, prior mean and covariance of the continuous distribution, with the new observation y_t. We omit details and the Kalman equations here, and refer interested readers to [7].

The Kalman filter is an extremely efficient algorithm. However, in the case of non-linear transformations it does not apply; good approximations are achieved by the *extended Kalman filter* (EKF) and the *unscented Kalman filter* (UKF) with the UKF generally dominating the EKF [22]. Rather than using the standard Kalman filter update to compute the posterior distribution, the UKF performs the following: Given an m-dimensional continuous space, $2m + 1$ *sigma points* are chosen based on the a-priori covariance (see [22] for details). The non-linear system equation is then applied to each of the sigma points, and the a-posteriori distribution is approximated by a Gaussian whose mean and covariance are computed from the sigma points. This unscented Kalman filter update yields an approximation of the posterior whose error depends on how different the true posterior is from a Gaussian. For linear and quadratic transformations, the error is zero.

Particle Filters

While the success of the above approaches depend on how strongly the belief state resembles a multivariate Gaussian, the *particle filter* (PF) [10] is applicable regardless of the underlying model. A particle filter is a Markov chain Monte Carlo algorithm that approximates the belief state using a set

1. For N particles $p^{(i)}$, $i = 1, \ldots, N$, sample discrete modes $z_0^{(i)}$, from the prior $P(Z_0)$.
2. For each particle $p^{(i)}$, sample $x_0^{(i)}$ from the prior $P(X_0|z_0^{(i)})$.
3. for each time-step t do
   (a) For each particle $p^{(i)} = (z_{t-1}^{(i)}, x(i)_{t-1})$ do
   i. Sample a new mode:

$$\hat{z}_t^{(i)} \sim P(Z_t|z_{t-1}^{(i)})$$

   ii. Sample new continuous parameters:

$$\hat{x}_t^{(i)} \sim P(X_t|\hat{z}_t^{(i)}, x_{t-1}^{(i)})$$

   iii. Compute the weight of particle $\hat{p}^{(i)}$:

$$w_t^{(i)} \leftarrow P(y_t|\hat{z}_t^{(i)}, \hat{x}_t^{(i)})$$

   (b) Resample N new samples $p^{(i)} = (z_t^{(i)}, x_t^{(i)})$ where: $P(p^{(i)} = \hat{p}^{(k)}) \propto w_t^{(k)} = \frac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}}$

Figure 1. The particle filtering algorithm.

of samples (particles), and keeps the distribution updated as new observations are made over time. The basic PF algorithm is shown in Figure 1. To update the belief distribution given a new observation, the algorithm operates in three steps as follows:

The Monte Carlo step: This step considers the evolution of the system over time. It uses the stochastic model of the system to generate a possible future state for each sample. In our hybrid model (and Figure 1), this is performed by sampling a discrete mode, and then the continuous state given the new mode.

The reweighting step: This corresponds to conditioning on the observations. Each sample is weighted by the likelihood of seeing the observations in the (updated) state represented by the sample. This step leads samples that predict the observations well to have high weight, and samples that are unlikely to generate the observations to have low weight.

The resampling step: To produce a uniformly weighted posterior, we then resample a set of uniformly weighted samples from the distribution represented by the weighted samples. In this resampling the probability that a new sample is a copy of a particular sample s is proportional to the weight of s, so high-weight samples may be replaced by several samples, and low-weight samples may disappear.

At any time t, the PF algorithm approximates the true posterior belief state given observations $y_{1:t}$ by a set of samples (or particles):

$$\begin{aligned} P(Z_t, X_t|y_{1:t}) &\approx \hat{P}(Z_t, X_t|y_{1:t}) \\ &= \frac{1}{N}\sum_{i=1}^{N} w_t^{(i)} \delta_{(Z_t,X_t)}((z_t^{(i)}, x_t^{(i)})) \end{aligned}$$

where $w_t^{(i)}$, $z_t^{(i)}$ and $x_t^{(i)}$ are weight, discrete mode and continuous parameters of particle $p^{(i)}$ at time t, N is the number of samples, and $\delta_x(y)$ denotes the Dirac delta function.

Particle filters have a number of properties that make them a desirable approximation algorithm for diagnosis. As we said above, unlike the Kalman filter, they can be applied to non-linear models with arbitrary prior belief distributions. They are also *contract anytime* algorithms, meaning that if you specify in advance how much computation time is available, a PF algorithm can estimate a belief distribution in the available time—by changing the number of samples, you trade off computation time for the quality of the approximation. In fact, the computational requirements of a particle filter depend *only* on the number of samples, not on the complexity of the model.

Unfortunately, as we said in the introduction, diagnosis problems have some characteristics that make standard particle filtering approaches less than ideal. In particular, on-board diagnosis for applications such as spacecraft and planetary rovers must be performed using very limited computational resources, and transitions to fault modes typically have very low probability of occurring. This second problem leads to a form of *sample impoverishment*, in which modes with a non-zero probability of being the actual state of the system contain no samples, and are therefore treated by the particle filter as having zero probability. This is particularly a problem for diagnosis, because these are exactly the states for which we are most interested in estimating the likelihood. There have been a few approaches to tackling this issue, most notably [5] and [17].

Another traditional problem of particle filters is that the number of samples needed to cope with high dimensional continuous state spaces is enormous. Especially in the case of high noise levels and widespread distributions, approximations via sampling do not yield good results. If it is possible to represent the continuous variables in a compact way, e.g. in the form of sufficient statistics, this generally helps by greatly reducing the number of particles needed. In the next section,

we introduce one instance of this, the highly efficient Rao-Blackwellized Particle Filter which only samples the discrete modes and propagates sufficient statistics for the continuous variables.

3. Risk-Sensitive Particle Filters

One way to think about Problem 1 in our list, the presence of very low-probability fault transitions, is in terms of risk. The reason these transitions are a serious concern in fault detection, but much less so in other applications of particle filters, is the fact that the low-probability transitions correspond to faults, the very thing we are most interested in detecting. The occurrence of faults has the potential for great risk to the rover, because a perfectly reasonable action in a nominal mode of rover behaviour may be catastrophic if an undetected fault has occurred.

RSPFs [20], [17] incorporate a model of cost when generating particles. This approach is motivated by the observation that the cost of not tracking hypotheses is related to risk. Not tracking a rare but risky state may have a high cost, whereas not tracking a rare but benign state may be irrelevant. Incorporating a cost model into particle filtering improves the tracking of states that are most critical to the performance of the robot.

Faults are low-probability, high-cost events. The classical particle filter generates particles proportional only to the posterior probability of an event. Monitoring a system to detect and identify faults based on a standard PF therefore requires a very large number of particles and is computationally expensive. RSPF generates particles by factoring in the cost. Since faults have a high cost, even though they have a low probability, a smaller number of particles than the PF may be used to monitor these events because the RSPF ensures particles will be generated to represent them.

The cost function assigns a real-valued cost to states and control. The control selected, given the exact state, results in the minimum cost. The approximate nature of the particle representation may result in sub-optimal control and hence increased cost. The goal of risk-sensitive sampling is to generate particles that minimize the cumulative increase in cost due to the approximate particle representation. This is done by modifying the classical particle filter to generate particles in a risk sensitive manner, where risk is defined as a function of the cost and is positive and finite. Given a suitable risk function $r(d)$, a risk-sensitive particle filter generates particles that are distributed according to the invariant distribution,

$$\gamma_t \; r(z_t) \; P(z_t, x_t | y_{1:t}) \tag{1}$$

where, γ_t is a normalization constant that ensures that equation (1) is a probability distribution. Instead of using just the posterior distribution to generate the particles, a product of the risk times the posterior is used. To achieve this, two modifications are made to the PF algorithm from Figure 1. First,

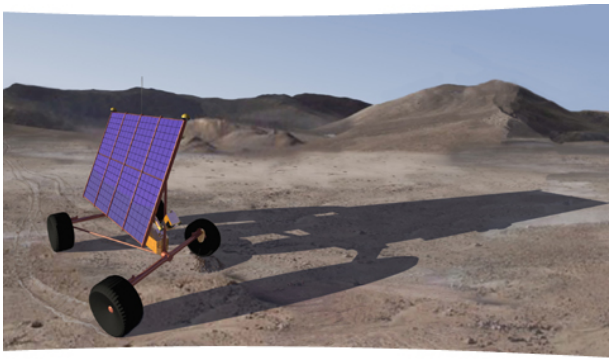


Figure 2. The Hyperion rover.

the initial set of particles (step 1) is generated from:

$$\gamma_0 \; r(z_0) \; P(Z_0)$$

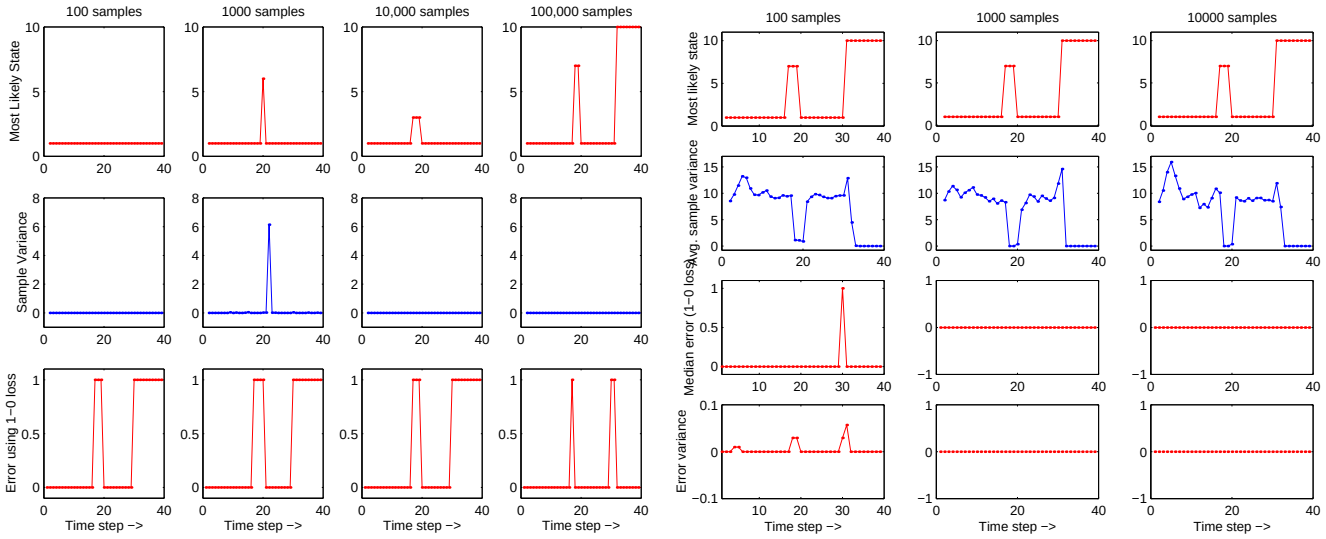
and the equation in step 3(a)iii is replaced with:

$$w_t^{(i)} = \frac{r(\hat{x}_t^{(i)})}{r(x_{t-1}^{(i)})} \; P(y_t | \hat{z}_t^{(i)}, \hat{x}_t^{(i)})$$

These simple modifications result in a particle filter with particles distributed according to $\gamma_t r(z_t) P(z_t, x_t | y_{1:t})$. The choice of risk function is important. For the experiments reported below, the risk function was computed heuristically. Thrun et. al. in [17] present a method for obtaining this risk function via a Markov decision process (MDP) that calculates the approximate future risk of decisions made in a particular state. Although we don't present it here, a similar approach to biasing the proposal distribution appears in [5].

Results: Risk-Sensitive Particle Filter

The Hyperion robot [23], figure 2, was the platform for the experiment with the RSPF. In a simulation of Hyperion, we explicitly introduced faults and recorded a sequence of controls and measurements that were then tracked by a RSPF and a standard PF. In the experiment the robot was driven with a variety of different control inputs in the normal operation mode. For this experiment, the measurements were the rover pose (x, y, θ) and steering angle. At the $17th$ timestep, wheel $\#3$ becomes stuck and locked against a rock. The wheel is then driven in the backward direction, fixing the problem. The robot returns to the normal operation mode and continues to operate normally until the gear on wheel $\#4$ breaks at the $30th$ time step. Figure 3 shows the results of tracking the state with a classical particle filter and with the RSPF; only the discrete state estimates are shown. Each column represents tests with different sample sizes (100, 1000, 10,000 and 100,000 samples respectively from left to right). We don't show results with the RSPF for 100,000 samples since the results are already accurate for smaller sample sizes. In each of these figures, along the x-axis is time. The top row shows the most likely discrete state estimate along the y-axis. The faults represented by the numbers are listed in the figure caption. Even with 100,000 particles in the classical filter, Figure 3(a), there



(a) Classical particle filter

(b) Risk-Sensitive PF

Figure 3. (a) Results with a simple particle filter. Here (1) normal, (2) wheel1 or wheel2 motor or gear broken, (3) wheel3 broken, (4) wheel4 broken, (5) wheel1 stuck, (6) wheel2 stuck, (7) wheel3 stuck, (8) wheel4 stuck, (9) wheel3 gear broken, (10) wheel4 gear broken (b) Results with a RSPF

is a slight lag in fault detection. With smaller sample sizes the most likely state estimate never transitions from the normal state. Occasionally particle do jump to fault states, (see column 2), but the sample immediately dies since it did not jump to the correct fault state. With the RSPF, Figure 3(b), the most likely states capture the true fault states for as few as a 100 samples. Row two in the figures shows the variance in the discrete state estimate. It contains the same information as in row one, but makes clear that with the classical filter the samples are in the nominal state almost all the time. Row 3 shows the mean squared error using 1-0 loss; it demonstrates that the RSPF has a low error with 100 samples and zero error with larger numbers of samples. The classical filter has maximum error whenever there is a fault. In addition, we also show the variance in the error in Figure 3(b) to demonstrate that the RSPF consistently provides estimates with low error.

4. Variable Resolution Particle Filter

As we said in the introduction, a well known problem with particle filters is that a large number of particles are often needed to obtain a reasonable approximation of the posterior distribution. For real-time state estimation maintaining such a large number of particles is typically not practical (Problems 2 and 3 on our list). In this section, we present the variable resolution particle filter [21], which addresses this problem by trading off bias and variance. The idea is based on the observation that the variance of the particle based estimate can be high with a limited number of samples, particularly when the process is not very stochastic and parts of the state space transition to other parts with very low, or zero, probability. Consider the problem of diagnosing locomotion faults on a robot. The probability of a stalled motor is low and wheels on the same side generate similar observations. Motors on any of the wheels may stall at any time. A particle filter that produces an estimate with a high variance is likely to result in identifying some arbitrary wheel fault on the same side, rather than identifying the correct fault.

The variable resolution particle filter introduces the notion of an abstract particle, in which particles may represent individual states or sets of states. With this method a single abstract particle simultaneously tracks multiple states. A limited number of samples are therefore sufficient for representing large state spaces. A bias-variance tradeoff is made to dynamically refine and abstract states to change the resolution, thereby abstracting a set of states and generalizing the samples or specializing the samples in the state into the individual states that it represents. As a result reasonable posterior estimates can be obtained with a relatively small number of samples. In the example above, with the VRPF the wheel faults on the same side of the rover would be aggregated together into an abstract fault. Given a fault, the abstract state representing the side on which the fault occurs would have high likelihood. The samples in this state would be assigned a high importance weight. This would result in multiple copies of these samples on resampling proportional to weight. Once there are sufficient particles to populate all the refined states represented by the abstract state, the resolution of the state would be changed to the states representing the individual wheel faults. At this stage, the correct hypothesis is likely to be included in this particle based approximation at the level of the individual states and hence the correct fault is likely to be detected.

For the variable resolution particle filter we need: (1) A variable resolution state space model that defines the relationship between states at different resolutions, (2) an algorithm for state estimation given a fixed resolution of the state space, (3) a basis for evaluating resolutions of the state space model, and (4) and algorithm for dynamically altering the resolution of the state space.

Variable resolution state space model

We could use a directed acyclic graph (DAG) to represent the variable resolution state space model, which would consider every possible combination of the (abstract) states to aggregate or split. But this would make our state space exponentially large. We must therefore constrain the possible combinations of states that we consider. There are a number of ways to do this. For the experiments in this paper we use a multi-layered hierarchy where each physical (non-abstract) state only exists along a single branch. Sets of states with similar state transition and observation models are aggregated together to create successively higher levels in the hierarchy. In addition to the physical state set $\{Z_k\}$, the variable resolution model, M consists of a set of abstract states $\{S_j\}$ that represent sets of states and or other abstract states.

$$S_j = \begin{cases} \{Z_k\} \\ \cup_i S_i \end{cases} \tag{2}$$

Figure 4(a) shows an arbitrary Markov model and figure 4(b) shows an arbitrary variable resolution model for 4(a). Figure 4(c) shows the model in 4(b) at a different resolution.

From the dynamics, $P(z_t|z_{t-1})$, and measurement probabilities $P(y_t|z_t)$, we compute the stationary distribution (Markov chain invariant distribution) of the physical states $\pi(Z_k)$ [1].

Belief state estimation at a fixed resolution

This section describes the algorithm for estimating a distribution over the state space, given a fixed resolution for each state, where different states may be at different fixed resolutions. For each particle in a physical state, a sample is drawn from the predictive model for that state $p(z_t|z_{t-1})$. It is then assigned a weight proportional to the likelihood of the measurement given the prediction, $p(y_t|z_t)$. For each particle in an abstract state, S_j, one of the physical states, z_t, that it represents in abstraction is selected proportional to the probability of the physical state under the stationary distribution, $\pi(Z_t)$. The predictive and measurement models for this physical state are then used to obtain a weighted posterior sample. The particles are then resampled proportional to their weight. Based on the number of resulting particles in each physical state a Bayes estimate with a Dirichlet(1) prior is obtained as follows:

$$\hat{P}(z_t|y_{1:t}) = \frac{n(z_t) + \pi(z_t)}{\mid N_t \mid +1}, \quad \sum_{z_t} \pi(z_t) = 1 \tag{3}$$

where, $n(z_t)$ represents the number of samples in the physical state z_t and $\mid N_t \mid$ represents the total number of particles in the particle filter. The distribution over an abstract state S_j at time t is estimated as:

$$\hat{P}(S_j|y_{1:t}) = \sum_{z_t \in S_j} \hat{P}(z_t|y_{1:t}) \tag{4}$$

Bias-variance tradeoff

The loss l, from a particle based approximation $\hat{P}(z_t|y_{1:t})$, of the true distribution $p(z_t|y_{1:t})$ is:

$$\begin{aligned} l &= E[P(z_t|y_{1:t}) - \hat{P}(z_t|y_{1:t})]^2 \\ &= \{P(z_t|y_{1:t}) - E[\hat{P}(z_t|y_{1:t})]\}^2 + \\ &\quad \{\hat{P}(z_t|y_{1:t})^2 - E[\hat{P}(z_t|y_{1:t})]^2\} \\ &= b(\hat{P}(z_t|y_{1:t}))^2 + v(\hat{P}(z_t|y_{1:t})) \end{aligned} \tag{5}$$

where, $b(\hat{P}(z_t|y_{1:t}))$ is the bias and $v(\hat{P}(z_t|y_{1:t}))$ is the variance.

The posterior belief state estimate from tracking states at the resolution of physical states introduces no bias. But the variance of this estimate can be high, specially with small sample sizes. An approximation of the sample variance at the resolution of the physical states may be computed as follows:

$$v(z_t) = \hat{P}(z_t|y_{1:t}) \ \frac{\hat{P}(z_t|y_{1:t}) \ [1 - \hat{P}(z_t|y_{1:t})]}{n(z_t) + \pi(z_t)} \tag{6}$$

The loss of an abstract state S_j, is computed as the weighted sum of the loss of the physical states $z_t \in S_j$, as follows[1]:

$$l(S_j) = \sum_{z_t \in S_j} \hat{P}(z_t|y_{1:t}) \ l(z_t) \tag{7}$$

The generalization to abstract states biases the distribution over the physical states to the stationary distribution. In other words, the abstract state has no information about the relative posterior likelihood, given the data, of the states that it represents in abstraction. Instead it uses the stationary distribution to project its posterior into the physical layer. The projection of the posterior distribution $\hat{P}(S_j|y_{1:t})$, of abstract state S_j, to the resolution of the physical layer $\tilde{P}(z_t|y_{1:t})$, is computed as follows:

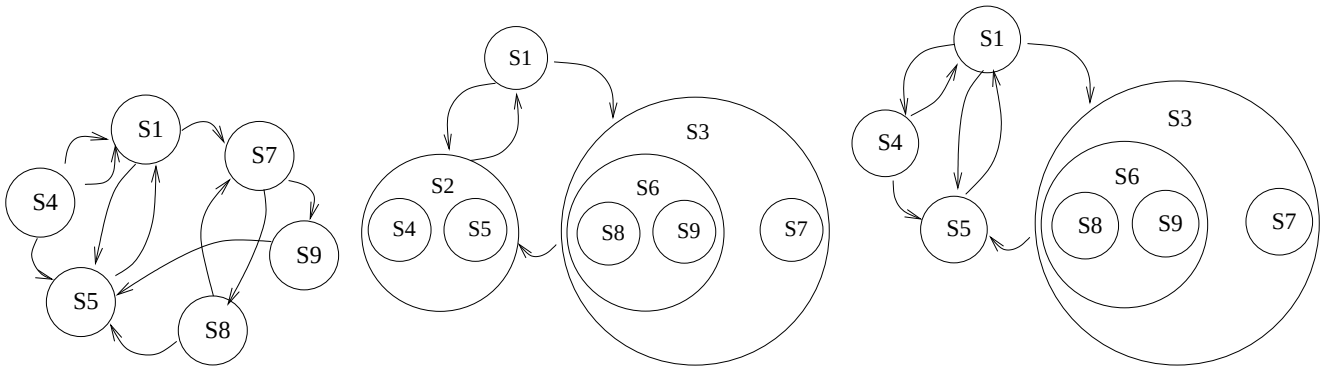
$$\tilde{P}(z_t|y_{1:t}) = \frac{\pi(z_t)}{\pi(S_j)} \ \hat{P}(S_j|y_{1:t}) \tag{8}$$

where, $\pi(S_j) = \sum_{z \in S_j} \pi(z)$.

As a consequence of the algorithm for computing the posterior distribution over abstract states described in the previous subsection, an unbiased posterior over physical states z_t is available at no extra computation, as shown in equation (3). The bias $b(S_j)$, introduced by representing the set of physical states $z_t \in S_j$, in abstraction as S_j is approximated as follows:

$$b(S_j) = \sum_{z_t \in S_j} \hat{P}(z_t|y_{1:t}) \ \ [\hat{P}(z_t|y_{1:t}) - \tilde{P}(z_t|y_{1:t})]^2 \tag{9}$$

[1]The relative importance/cost of the physical states may also be included in the weight



(a) Example (b) Variable resolution model (c) S2 at a finer resolution

Figure 4. (a) Arbitrary Markov model (b) Arbitrary variable resolution model corresponding to the Markov model in (a). Circles that enclose other circles represent abstract states. S2 and S3 and S6 are abstract states. States S2, S4 and S5 form one abstraction hierarchy, and states S3, S6, S7, S8 and S9 form another abstraction hierarchy. The states at the highest level of abstraction are S1, S2 and S3. (c) The model in (b) with states $S4$ and $S5$ at a finer resolution.

It is the weighed sum of the squared difference between the unbiased posterior $\hat{P}(z_t|y_{1:t})$, computed at the resolution of the physical states and the biased posterior $\tilde{P}(z_t|y_{1:t})$, computed at the resolution of abstract state S_j.

An approximation of the variance of abstract state S_j is computed as a weighted sum of the projection to the physical states as follows:

$$v(S_j) = \sum_{z_t \in S_j} \hat{P}(z_t|y_{1:t}) \left[\frac{\pi(z_t)}{\pi(S_j)}\right]^2 \frac{\hat{P}(S_j|y_{1:t})[1 - \hat{P}(S_j|y_{1:t})]}{n(S_j) + \pi(S_j)}$$

The loss from tracking a set of states $z_t \in S_j$, at the resolution of the physical states is thus:

$$l_p = 0 + \sum_{z_t \in S_j} v(z_t) \tag{10}$$

The loss from tracking the same set of states in abstraction as S_j is:

$$l_a = b(S_j) + v(S_j) \tag{11}$$

There is a gain in terms of reduction in variance from generalizing and tracking in abstraction, but it results in an increase in bias. Here, a tradeoff between bias and variance refers to the process of accepting a certain increase in one term for a larger reduction in the other and hence in the total error.

Dynamically varying resolution

The variable resolution particle filter uses a bias-variance tradeoff to make a decision to vary the resolution of the state space. A decision to abstract to the coarser resolution of abstract state S_j, is made if the state space is currently at the resolution of states S_i, and the combination of bias and variance in abstract state S_j, is less than the combination of bias an variance of all its children S_i, as shown below:

$$b(S_j) + v(S_j) \leq \sum_{S_i \in \{children(S_j)\}} [b(S_i) + v(S_i)] \tag{12}$$

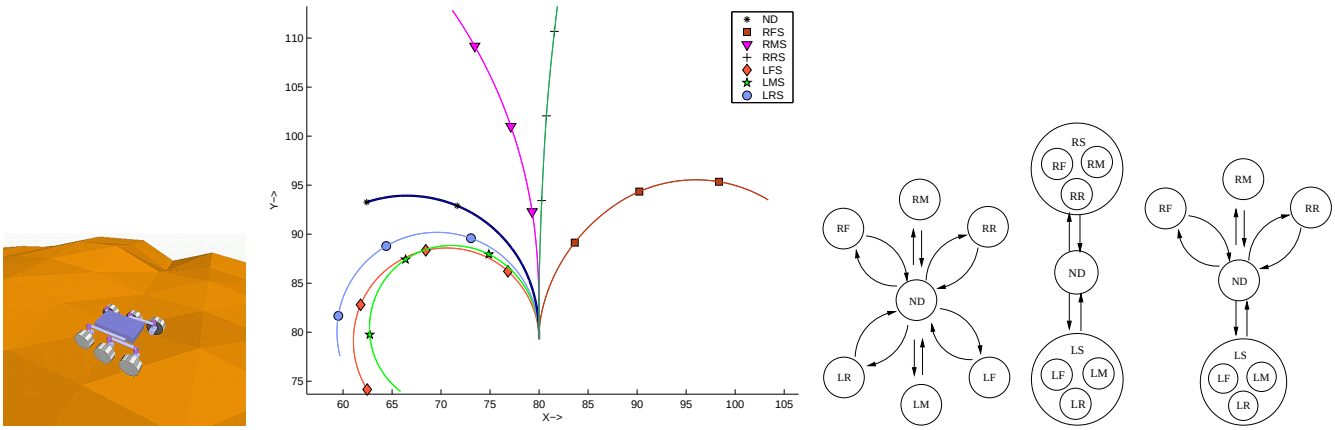
On the other hand if the state space is currently at the resolution of abstract state S_j, and the reverse of equation (12) is true, then a decision to refine to the finer resolution of states S_i is made. The resolution of a state is left unaltered if its bias-variance combination is less than its parent and its children. To avoid hysteresis, all abstraction decisions are considered before any refinement decisions.

Each time a new measurement is obtained the distribution of particles over the state space is updated. Since this alters the bias and variance tradeoff, the states explicitly represented at the current resolution of the state space are each evaluated for gain from abstraction or refinement. Any change in the current resolution of the state space is recursively evaluated for further change in the same direction.

Results: Variable Resoluton Particle Filter

The problem domain for our experiments on the variable resolution PF involves diagnosing locomotion faults in a physics based simulation of a six wheel rover. Figure 5(a) shows a snapshot of the rover in the Darwin2K [12] simulator.

The experiment is formulated in terms of estimating discrete fault and operational modes of the robot from continuous control inputs and noisy sensor readings. The discrete state, x_t, represents the particular fault or operational mode. The continuous variables, z_t, provide noisy measurements of the change in rover position and orientation. The particle set P_t therefore consists of N particles, where each particle $x_t^{[i]}$ is a hypothesis about the current state of the system. In other words, there are a number of discrete fault and operational states that a particle may transition to based on the transition



(a) Rover in D2K (b) Change in trajectory (c) (d) (e)

Figure 5. (a) Snapshot from the dynamic simulation of the six wheel rocker bogie rover in the simulator, (b) An example showing the normal trajectory (ND) and the change in the same trajectory with a fault at each wheel. (c) Original discrete state transition model. The discrete states are: Normal driving (ND), right and left, front, middle and rear wheel faulty (RF, RM, RR, LF, LM, LR) (d) Abstract discrete state transition model. The states, RF, RM and RR have been aggregated into the Right Side wheel faulty states and similarly LF, LM and LR into Left Side wheel faulty states (RS and LS). (e) State space model where RS has been refined. All states have self transitions that have been excluded for clarity.

model. Each discrete fault state has a different observation and predictive model for the continuous dynamics. The probability of a state is determined by the density of samples in that state.

The Markov model representing the discrete state transitions consists of 7 states. As shown in figure 5(c) the normal driving (ND) state may transition back to the normal driving state or to any one of six fault states: right front (RF), right middle (RM), right rear (RR), left front (LF), left middle (LM) and left rear (LR) wheel stuck. Each of these faults cause a change in the rover dynamics, but the faults on each side (right and left), have similar dynamics.

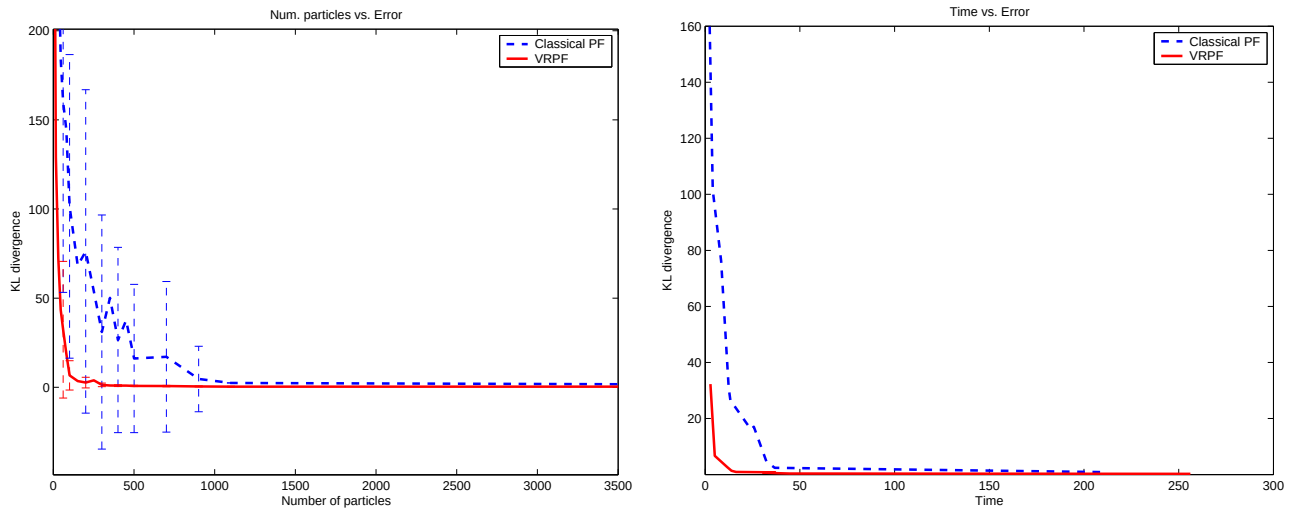
Given that the three wheels on each side of the rover have similar dynamics, we constructed a hierarchy that clusters the fault states on each side together. Figure 5(d) shows this hierarchical model, where the abstract states right side fault (RS), and left side fault (LS) represent sets of states $\{$RF, RM, RR$\}$ and $\{$LF, LM, LR$\}$ respectively. The highest level of abstraction therefore consists of nodes $\{$ND, RS, LS$\}$. Figure 5(e) shows how the state space in figure 5(d) would be refined if the bias in the abstract state RS given the number of particles outweighs the reduction in variance over the specialized states RF, RM and RR at a finer resolution.

When particle filtering is performed with the variable resolution particle filter, the particles are initialized at the highest level in the abstraction hierarchy,i.e. in the abstract states ND, RS and LS. Say a RF fault occurs, this is likely to result in a high likelihood of samples in RS. These samples will multiply which may then result in the bias in RS exceeding the reduction in variance in RS over RF, RM and RR thus favoring tracking at the finer resolution. Additional observations should then assign a high likelihood to RF.

The model is based on the real-world and is not very stochastic. It does not allow transitions from most fault states to other fault states. For example, the RF fault does not transition to the RM fault. This does not exclude transitions to multiple fault states and if the model included multiple faults, it could still transition to a "RF and RM" fault, which is different from a RM fault. Hence, if there are no samples in the actual fault state, samples that end up in fault states with dynamics that are similar to the actual fault state may end up being identified as the fault state. The hierarchical approach tracks the state at an abstract level and does not commit to identifying any particular specialized fault state until there is sufficient evidence. Hence it is more likely to identify the correct fault state.

Figure 6(a) shows a comparison of the error from monitoring the state using a classical particle filter that tracks the full state space, and the VRPF that varies the resolution of the state space. The X axis shows the number of particles used, the Y axis shows the KL divergence from an approximation of the true posterior computed using a large number of samples. 1000 samples were used to compute an approximation to the true distribution. The Kullback-Leibler (KL) divergence [11] is computed over the entire length of the data sequence and is averaged over multiple runs over the same data set [2]. The data set included normal operation and each of the six faults.

[2] The results are an average over 50 to 5 runs with repetitions decreasing as the sample size was increased.



(a) (b)

Figure 6. Comparison of the KL divergence from the true distribution for the classical particle filter and the VRPF, against (a) number of particles used, (b) wall clock time.

Figure 6(a) demonstrates that the performance of the VRPF is superior to that of the classical filter for small sample sizes. In addition figure 6(b) shows the KL divergence along the Y axis and wall clock time along the X axis. Both filters were coded in matlab and share as many functions as possible.

The Variable Resolution Particle filter has also been extended to use lookahead using UKFs [9]. Lookahead requires computing a UKF for every possible transition to a fault or nominal state at each instance in time. The VRPF introduced the notion of abstract states that may represent sets of states. There are fewer transitions between states when they are repented in abstraction. We show that the VRPF in conjunction with a UKF proposal improves performance and may potentially be used in large state spaces [19].

5. Rao-Blackwellized Particle Filters

Much recent work on *Rao-Blackwellized Particle Filtering* (RBPF) [2], [14] has focused on combining PFs and KFs for tracking linear multi-modal systems with Gaussian noise. This approach is very effective at tracking system state using a very small number of samples (Problem 2 on our list). In this kind of model, the belief state is a mixture of Gaussians. Rather than sampling a complete system state, in RBPF for hybrid systems, one combines a Particle Filter that samples the discrete modes z_t, and a Kalman Filter for each discrete mode $z_t \in Z$ that propagates sufficient statistics $(\mu_t^{(i)}, \Sigma_t^{(i)})$ for the continuous parameters x_t. The algorithm is shown in Figure 7. At each time-step t, first, the discrete mode is sampled according to the transition prior. Then, for each particle $p^{(i)}$ a Kalman filter is called to compute the prior mean $\hat{y}_{t|t-1}^{(i)}$ and covariance $\hat{S}_t^{(i)}$ of the observation and update the mean $\mu_t^{(i)}$ and covariance $\Sigma_t^{(i)}$ for the continuous parameters. The variable $\theta(z_t^{(i)})$ denotes the parameters of the Kalman Filter belonging to mode $z_t^{(i)}$. Finally, the particle weight is computed as the observation probability $P(y_t|\hat{y}_{t|t-1}^{(i)}, \hat{S}_t^{(i)})$ of y_t given the prior observation mean and covariance. As in regular Particle Filtering, a resampling step is necessary to prevent particle impoverishment.

As shown in [14], it is possible in Rao-Blackwellized Particle Filtering to sample the discrete modes directly from the posterior. It is also possible to resample *before* the transition according to the expected posterior weight distribution such that those particles get multiplied which are likely to transition to states of high confidence. These improvements result in an even more efficient algorithm called RBPF2 [14].

Non-Linear Estimation

Since RBPF uses a KF for its continuous state estimation, it is restricted to linear problems with Gaussian noise. Many of the problems we are interested in do not have these properties. To overcome this, we propose the *Gaussian particle filter* (GPF). In general hybrid systems, there is no tractable closed-form solution for the continuous variables, so we cannot maintain sufficient statistics with every sample. It is however possible to propagate an approximation of the continuous variables. We sample the mode as usual and for every particle update a Gaussian approximation of the continuous parameters using an unscented Kalman filter. Since the unscented Kalman filter only approximates the true posterior distribution, the GPF is a biased estimator in non-linear models; however, by not sampling the continuous state, we greatly reduce the estimator's variance.

1. For N particles $p^{(i)}$, $i = 1, \ldots, N$, sample discrete modes $z_0^{(i)}$, from the prior $P(Z_0)$.
2. For each particle $p^{(i)}$, set $\mu_0^{(i)}$ and $\Sigma_0^{(i)}$ to the prior mean and covariance in state $z_0^{(i)}$.
3. For each time-step t do
   (a) For each $p^{(i)} = (z_{t-1}^{(i)}, \mu_{t-1}^{(i)}, \Sigma_{t-1}^{(i)})$ do
   i. Sample a new mode:

$$\hat{z}_t^{(i)} \sim P(Z_t | z_{t-1}^{(i)})$$

   ii. Perform Kalman update using parameters from mode $\hat{z}_t^{(i)}$:

$$(\hat{y}_{t|t-1}^{(i)}, \hat{S}_t^{(i)}, \hat{\mu}_t^{(i)}, \hat{\Sigma}_t^{(i)}) \leftarrow KF(\mu_{t-1}^{(i)}, \Sigma_{t-1}^{(i)}, y_t, \theta(z_t^{(i)}))$$

   iii. Compute the weight of particle $\hat{p}^{(i)}$:

$$w_t^{(i)} \leftarrow P(y_t | \hat{y}_{t|t-1}^{(i)}, \hat{S}^{(i)}) = N(y_t; \hat{y}_{t|t-1}^{(i)}, \hat{S}^{(i)}).$$

   (b) Resample as in step 3.(b) of the PF algorithm (see Figure 1).

Figure 7. The RBPF algorithm.

The GPF algorithm is very similar to the RBPF algorithm presented in Figure 7. In both of these algorithms particle $p^{(i)}$ represents the continuous variables with a multivariate Gaussian $N(\mu_t^{(i)}, \Sigma_t^{(i)})$. In the case of linear models and RBPF, this Gaussian is a sufficient statistic, in the case of non-linear models and GPF, it is an approximation. In the algorithm, the only change is in line 3.(a)ii of Figure 7, which is replaced by:

3.a(ii) Perform an unscented Kalman update using parameters from mode $\hat{z}_t^{(i)}$:

$$(\hat{y}_{t|t-1}^{(i)}, \hat{S}_t^{(i)}, \hat{\mu}_t^{(i)}, \hat{\Sigma}_t^{(i)}) \leftarrow UKF(\mu_t^{(i)}, \Sigma_t^{(i)}, y_t, \theta(z_t^{(i)}))$$

This change is due to the non-linearity of transition and/or observation function. A Kalman update is simply not possible, but a good approximation is achieved with an unscented Kalman filter. The approximation of continuous variables in the GPF is a mixture of Gaussians rather than the set of samples as in a PF. Since the expressive power of every particle is higher, fewer particles are needed to achieve the same approximation accuracy. This more than offsets the small additional computational cost per sample. Furthermore, this compact approximation is likely to scale smoothly with an increase in dimensionality.

Lookahead for Low-Probability Transitions

Like RBPF, the GPF can be improved by sampling directly from the posterior distribution and resampling *before* the transition. This allows us to improve the probability of having a sample follow a low-probability transition (Problem 1) because the probability of such a sample is based on the posterior likelihood (the observation is taken into account) of the transition, rather than the prior. We call the resulting algorithm GPF2 and detail it in Figure 8. For each particle, before actually sampling a discrete mode, we look at each possible mode m, update our approximations of the continuous parameters assuming we had sampled m, and compute the observation likelihood for those approximations. This and the transition prior give the posterior probability of transitioning to m. Then for each particle we sample a new discrete mode from the posterior we computed for it.

At each time-step t, for every particle $p^{(i)}$, first we enumerate each possible successor mode m, i.e. each mode $m \in Z$ such that $P(m|z_{t-1}^{(i)}) > 0$. For each m, we do an unscented Kalman update, and compute analytically the observation likelihood $P(y_t|m, \mu_{t-1}^{(i)}, \Sigma_{t-1}^{(i)}) = P(y_t|y_{t|t-1}^{(i,m)}, S_t^{(i,m)})$. Then, we compute the unnormalized posterior probability $Post(i,m)$ of particle $p^{(i)}$ transitioning to m; this is the product of the transition prior to m and the observation likelihood in m. Next we compute the weight of each particle $\hat{p}^{(i)}$ as the sum of the posterior probabilities of it's successor modes and resample N particles according to this weight distribution. Note, that $Post(i,m)$, $\mu_t^{(i,m)}$ and $\Sigma_t^{(i,m)}$ also need to be resampled, i.e. when particle $p^{(i)}$ is sampled to be particle $\hat{p}^{(k)}$, then $Post(i,m) \leftarrow \widehat{Post}(k,m)$, $\mu_t^{(i,m)} \leftarrow \hat{\mu}_t^{(k,m)}$ and $\Sigma_t^{(i,m)} \leftarrow \hat{\Sigma}_t^{(k,m)}$ for all m.

Finally, for every particle $p^{(i)}$, a successor mode m is sampled according to the posterior probability; this mode is used as $z_t^{(i)}$; $\mu_t^{(i)}$ and $\Sigma_t^{(i)}$ are set to the already computed value $\mu_t^{(i,m)}$ and $\Sigma_t^{(i,m)}$.

GPF2 only differs from the RBPF2 algorithm in that it is calling an unscented Kalman filter update instead of a Kalman update due to the non-linear character of the transformations. It is a very efficient algorithm for state estimation on non-linear models with transition and observation functions that transform a Gaussian distribution to a distribution that's close to a Gaussian. Very low fault priors are handled especially gracefully by GPF2 since it samples the discrete modes from

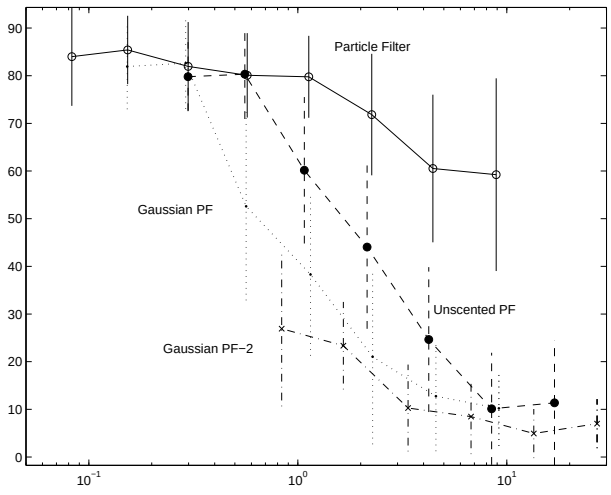


Figure 9. Performance for the GPF, the GPF when sampling from the posterior, the UPF, and traditional particle filters. The x-axis is CPU time, the y-axis is error rate (percentage of the time the most probable state according to the algorithm is not the true state of the system). Estimation based on 25 runs.

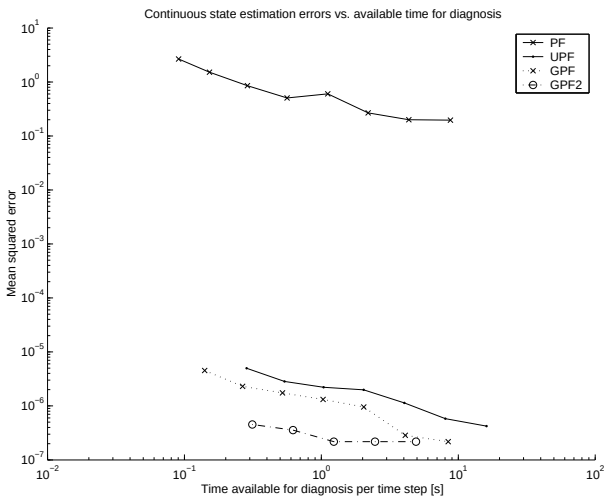


Figure 10. Mean squared errors of the four algorithms, averaged over 50 runs. Note the logarithmic scale for both the X- and Y- axes. At real time (ca. 1/3s per time step), the MSE of GPF2 is about six times lower than of GPF, ten times lower than that for UPF and 10^6 times lower than for PF.

their true posterior distribution. When there is strong enough evidence the fault will be detected regardless of how low the prior is.

Results: The Gaussian Particle Filter

We use a simple model of the suspension system of the K-9 rover (Figure 11) at NASA Ames Research Center. K-9 is a six wheeled rover with a rocker-bogey suspension, and we model the suspension's response to driving over rocks and other obstacles to anticipate situations where the rover's scientific instruments could collide with an obstacle, or where the rover could become "high-centered" on a rock. The model has six discrete modes and six continuous variables, two of which are observable. The continuous parameters follow non-linear trajectories in three of the modes.

Figure 9 shows the rate of state estimation errors for the GPF, GPF2 and traditional particle filters, as well as the unscented particle filter (discussed below) on data generated from the model. The diagnoses are taken to be the maximum a posteriori (MAP) estimate for the discrete modes; a discrepancy between this MAP estimate and the real discrete mode is an error. Figure 9 shows the error rates ($\frac{\#diagnosis\ errors}{\#time\ \ steps}$) for different numbers of samples; the x-axis is the CPU time. The graph shows that GPF is a better approximation than PF given the same computing resources, particularly as the number of samples increases and the discrete states become adequately populated with samples. GPF2 is considerably slower per sample but its approximation is superior to PF or GPF.

We are also interested in diagnosing the continuous parameters of the system. Figure 10 shows the mean squared error (MSE) of the algorithms on artificial data where ground truth is available.

Figure 11. NASA Ames' K-9 rover.

Finally, we applied GPF to real data from the K-9 rover. In Figure 12, we show the two observed variables, differential angle (Y2) and bogey angle (Y1) as well as the discrete mode estimates PF and GPF2 yield on this data. State 1 represents flat driving, state 2 driving over a rock with the front wheel, state 3 with the middle wheel and state 4 with the rear wheel. State 5 represents the rock being between the front and the middle wheel and state 6 between the middle and the rear wheel. Both filters successfully detect the rocks, but the GPF2 detects all of the rocks before PF detects them. For the third rock in the data, GPF2 correctly identifies that the back wheel passed over the rock, while the particle filter only tracks the first two wheels. Again, we only show the most

1. For N particles $p^{(i)}$, $i = 1, \ldots, N$, sample discrete modes $z_0^{(i)}$, from the prior $P(Z_0)$.
2. For each particle $p^{(i)}$, set $\mu_0^{(i)}$ and $\Sigma_0^{(i)}$ to the prior mean and covariance in state $z_0^{(i)}$.
3. For each time-step t do

(a) For each $p^{(i)} = (z_{t-1}^{(i)}, \mu_{t-1}^{(i)}, \Sigma_{t-1}^{(i)})$ do

i. For each possible successor mode $m \in succ(z_{t-1}^{(i)})$ do

A. Perform unscented Kalman update using parameters from mode m:

$$(\hat{y}_{t|t-1}^{(i,m)}, \hat{S}_t^{(i,m)}, \hat{\mu}_t^{(i,m)}, \hat{\Sigma}_t^{(i,m)}) \leftarrow UKF(\mu_{t-1}^{(i)}, \Sigma_{t-1}^{(i)}, y_t, \theta(m))$$

B. Compute posterior probability of mode m as:

$$\begin{aligned} \widehat{Post}(i,m) &\leftarrow P(m|z_{t-1}^{(i)}, y_t) \\ &= P(m|z_{t-1}^{(i)}) N(y_t; y_{t|t-1}^{(i,m)}, S_{t|t-1}^{(i,m)}). \end{aligned}$$

ii. Compute the weight of particle $\hat{p}^{(i)}$: $w_t^{(i)} \leftarrow \sum_{m \in succ(z_{t-1}^{(i)})} \widehat{Post}(i,m)$

(b) Resample as in step 2.(b) of the PF algorithm (see Figure 1) (also resample $Post$, μ_t and Σ_t).

(c) For each particle $p^{(i)}$ do

i. Sample a new mode:

$$m \sim P(Z_t|z_{t-1}^{(i)}, y_t)$$

ii. Set $z_t^{(i)} \leftarrow m$, $\mu_t^{(i)} \leftarrow \mu_t^{(i,m)}$ and $\Sigma_t^{(i)} \leftarrow \Sigma_t^{(i,m)}$.

Figure 8. The GPF2 algorithm.

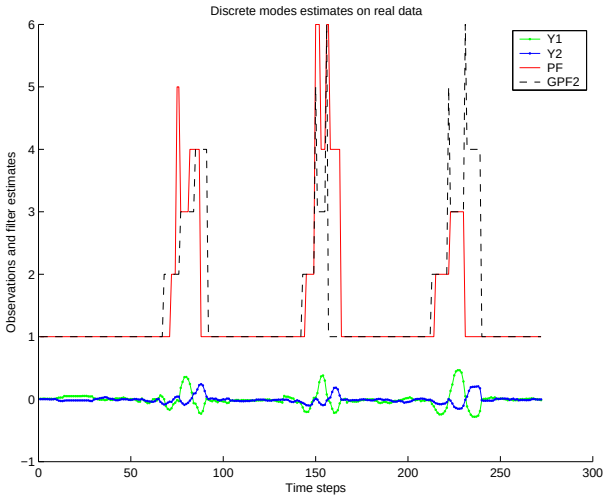


Figure 12. Discrete mode estimates on real data. Y1 is the bogey angle, Y2 is the rocker angle, and PF and GPF2 show the most probable state according to the particle filter and Gaussian particle filter with lookahead algorithms respectively.

probable mode at each time-step in the figure.

As well as a standard particle filter, we compare our results with the *unscented particle filter* (UPF) of [18]. The GPF and UPF have a number of similarities. Both use a set of particles each of which performs an unscented Kalman update at every time step. In UPF, the Kalman update approximation $N(mu_t, \Sigma_t)$ of the posterior is used as a proposal for the particle filter, in GPF this approximation is used as the filter result.

In our experiments there is little difference between the results of GPF and UPF. GPF is generally faster by a constant factor since it does not need to sample the continuous state, and the weight computation is faster. We would expect the UPF to yield better results when the shape of the posterior distribution is very different from a Gaussian and would expect the GPF to do better when there is a big posterior covariance Σ_t such that the sampling introduces high variance on the estimate. In this case, the UPF will need more particles to yield the same results. Since neither of these conditions applies in our domain, both algorithms show similar performance, with GPF being slightly faster.

6. Future Work

The three algorithms we have presented here are in many ways complementary. For example, we can easily imagine applying the GPF approach with the hiererchy of abstract states introduced in the variable resolution algorithm. Similarly, we could imagine using the risk-sensitive algroithm to bias the transition probabilities, although for the GPF2 with lookahead, this may not be necessary except in circumstances where the immediate observation does not indicate that the fault has occurred. We plan to begin integrating ideas from all three algorithms in the near future.

Another approach that is related to the variable resolution algorithm is the structured particle filter introduced in [16]. Here the samples themselves are split between states, giving

a similar performance boost and tackling Problem 3, the high-dimensional state space. We are currently in the process of fitting this algorithm into the GPF. This also makes progress on the problem of system-level diagnosis. Traditional diagnosis algorithms are component-based, with an emphasis on properties such as "no function in structure", meaning that subsystem models can be composed to make larger systems with ease. The challenge is to achieve these goals for hybrid models as well, and the structured particle filter approach should be a very effective way to exploit the structure that such a model implies.

Finally, we have only just begun testing these algorithms on real rover problems. Mostly this has been due to a lack of sensing on-board the available rover platforms, and on a lack of models and data for rover faults. However, if these algorithms are ever going to be used on a real rover mission to Mars or elsewhere, considerable testing on rovers must be done to demonstrate that the algorithms are robust, that their computational needs are not unreasonable, and that they scale to systems as complex as the whole rover. While this project has made progress on many of these goals, there is much still to be done, and in particular, testing on rovers running for long enough periods that faults do occur, is an important priority, and something we fully intend to do in the future.

REFERENCES

[1] E. Behrends. *Introduction to Markov Chains with Special Emphasis on Rapid Mixing*. Vieweg Verlag, 2000.

[2] Nando de Freitas. Rao-Blackwellised particle filtering for fault diagnosis. In *IEEE Aerospace*, 2002.

[3] Johann de Kleer and Brian C. Williams. Diagnosing multiple faults. In Matthew L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pages 372–388. Morgan Kaufmann, Los Altos, California, 1987.

[4] Johann de Kleer and Brian C. Williams. Diagnosis with behavioral modes. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1324–1330, 1989.

[5] Richard Dearden and Dan Clancy. Particle filters for real-time fault detection in planetary rovers. In *Proceedings of the Thirteenth International Workshop on Principles of Diagnosis*, pages 1–6, Semmering, Austria, 2002.

[6] A. Doucet. On sequential simulation-based methods for Bayesian filtering. Technical Report CUED/F-INFENG/TR.310, Department of Engineering, Cambridge University, 1998.

[7] M. S. Grewal and A. P. Andrews. *Kalman Filtering: Theory and Practice*. Prentice Hall, 1993.

[8] Michael W. Hofbaur and Brian C. Williams. Hybrid diagnosis with unknown behaviour modes. In *Proceedings of the Thirteenth International Workshop on Principles of Diagnosis*, pages 97–105, Semmering, Austria, 2002.

[9] Frank Hutter and Richard Dearden. The gaussian particle filter for diagnosis of non-linear systems. In *Proceedings of the Fourteenth International Workshop on the Principles of Diagnosis*, Washington, DC, 2003.

[10] M. Isard and A. Blake. CONDENSATION: Conditional density propagation for visual tracking. *International Journal of Computer Vision*, 1998.

[11] S. Kullback. *Information Theory and Statistics*. Dover, 1968.

[12] P. C. Leger. *Darwin2K: An evolutionary approach to automated design for robotics*. Kluwer Academic Publishers, 2000.

[13] Uri Lerner, Ron Parr, Daphne Koller, and Gautam Biswas. Bayesian fault detection and diagnosis in dynamic systems. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 2000.

[14] Ruben Morales-Menendez, Nando de Freitas, and David Poole. Real-time monitoring of complex industrial processes with particle filters. In *Neural Information Processing Systems (NIPS)*, 2002.

[15] Nicola Muscettola, Pandu Nayak, Barney Pell, and Brian Williams. Remote agent: To boldly go where no ai system has gone before. *Artificial Intelligence*, 103(1–2), August 1998.

[16] Brenda Ng, Leonid Peshkin, and Avi Pfeffer. Factored particles for scalable monitoring. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, Edmonton, 2002.

[17] Sebastian Thrun, John Langford, and Vandi Verma. Risk sensitive particle filters. In *Neural Information Processing Systems (NIPS)*, December 2001.

[18] Rudolph van der Merwe, Nando de Freitas, Arnaud Doucet, and Eric Wan. The unscented particle filter. In *Advances in Neural Information Processing Systems 13*, Nov 2001.

[19] V. Verma, G. Gordon, and R. Simmons. Efficient monitoring for planetary rovers. In *International Symposium on Artificial Intelligence and Robotics in Space*, 2003.

[20] V. Verma, J. Langford, and R. Simmons. Non-parametric fault identification for space rovers. In *International Symposium on Artificial Intelligence and Robotics in Space (iSAIRAS)*, 2001.

[21] V. Verma, S. Thrun, and R. Simmons. Variable resolution particle filter. In *International Joint Conference of Artificial Intelligence*, 2003.

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

[23] D. S. Wettergreen, M.B. Dias, B. Shamah, J. Teza, P. Tompkins, C. Urmson, M.D. Wagner, and W.L. Whittaker. First experiment in sun-synchronous exploration.

In *International Conference on Robotics and Automation*, 2002.

[24] Brian C. Williams and P. Pandurang Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference*, pages 971–978, Portland, Oregon, 1996. AAAI Press / The MIT Press.

Richard Dearden *is the PI of the Mars Technology Program task "Real-Time Fault Detection and Situational Awareness for Rovers." He is also currently acting group lead for the Model-based Diagnosis and Recovery group at NASA Ames Research Center. Richard has a Ph.D. and M.Sc. from the University of British Columbia, and a B.Sc. and B.Sc.(Hons.) from Te Whare Wananga o te Upoko o te ika a Maui/Victoria University of Wellington, New Zealand. He has published numerous papers in a number of areas, including decision-theoretic planning, reinforcement learning, and diagnosis.*

Frank Hutter *is a graduate student in the Department of Computer Science at Darmstadt University of Technology, Germany. He received German degrees ("Vordiplome") closely related to the BSc in computer science in 2000, as well as in commercial information technology in 2001. In the academic year 2001/02, he attended the University of British Columbia (UBC) as a visiting graduate student, and upon concluding his studies in Darmstadt with the master's degree (German "Diplom") in Spring 2004, he will join the PhD programme of UBC.*
He is a fellow of the Summer Student Research Program at NASA Ames Research Center, where he spent the summers of 2001 and 2002, researching on algorithms for Probabilistic Inference. Apart from this, his research focusses on the development of Stochastic Local Search algorithms and their application to problems in a variety of fields.

Reid Simmons *is a Research Professor in the School of Computer Science at Carnegie Mellon University. His research interests focus on developing reliable, highly autonomous systems (especially mobile robots) that operate in rich, uncertain environments. In particular, he is interested in architectures for autonomy that combine deliberative and reactive behavior, reliable execution monitoring and error recovery, multi-robot coordination, probabilistic and symbolic planning, formal verification of autonomous systems, and human-robot social interaction.*

Sebastian Thrun *is the director of the AI Lab at Stanford University and an associate professor of computer science. He pursues research in probabilistic reasoning, robotics, and machine learning with enthusiasm.*

Vandi Verma *is a Ph.D. student in Robotics at Carnegie Mellon University. Her primary research interests are in artificial intelligence, machine learning and robotics, with a focus on tractable methods for control and state estimation. The robots she has worked on have included Nomad (CMU, Antarctice meteorite search), Lama (LAAS, France), Bullwinkle (CMU, long-range Mars navigation), Hyperion (CMU, Sun-synchronous navigation), K8 (NASA Ames), and the robotic astrobiologist (CMU, life in the Atacama project).*

Thomas Willeke *is currently a member of the Model-based Diagnosis and Recovery Group, Computational Sciences Division at NASA Ames Research Center, where he is researching modeling of complex systems and hybrid diagnosis algorithms for those models. Currently he is applying these techniques to rovers for Martian exploration and to Martian subsurface drilling systems. Thomas earned a B.A. (1997) in Symbolic Systems and a M.S. (1998) in Computer Science from Stanford University. From there Thomas became the Technical Lead and Robot Personality Engineer at Mobot Inc. (Pittsburgh, PA), a small robotics startup which built fully autonomous robotic tour guides for the museum market. The four robots which were built there accumulated over seven years of autonomous unsupervised interaction with the general public. The work at Mobot instilled a lasting interest in long term reliability of autonomous robots.*