Non-Parametric Fault Identification for Space Rovers

Vandi Verma, John Langford, Reid Simmons
{vandi, jcl, reids}@cs.cmu.edu
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh PA 15213

Abstract

Autonomous fault detection is prerequisite for autonomous system repair which is of great value for spacecraft where human intervention is expensive, slow, unreliable, and occasionally impossible. We present a method which successfully achieves autonomous fault detection in simulation. The approach uses a nonparametric estimate of the system state updated based upon sensor measurements. Our system does state estimation using a decision-theoretic generalization of particle filters which takes into account the difference in utility of fault detection vs. fault nondetection.

1 Introduction

In the last ten years, three space missions have disappeared before reaching Mars – Mars Observer, Mars Climate Orbiter and Mars Polar Lander.

On August 21, 1993 the transmitters on the Mars Observer were turned off during the final approach to Mars to protect the components against shock from the pressurization sequence. After the transmitter was turned off the tanks were supposed to be pressurized and then the transmitters turned back on and communications with Earth resumed, but no further signals were ever received on Earth. The hypothesis is that a small amount of nitrogen tetroxide may have leaked through the check valves during the 11 month voyage to Mars and condensed in the pressurization lines. During pressurization, the oxidant would have mixed with the monomethylhydrazine fuel, causing combustion and rupture of the fuel lines. The resultant high-pressure expulsion of gases through the rupture would have started the spacecraft spinning uncontrollably and making communication with Earth impossible.

Mars Climate Orbiter, the next of NASA's Mars missions, was supposed to enter orbit around Mars on September 23rd, but ground controllers did not receive a signal as scheduled. The spacecraft was supposed to pass about 150 kilometers above the surface of Mars and use the atmosphere to slow itself down enough to enter orbit. Data from the stations tracking the spacecraft shows that the spacecraft passed only about 60 kilometers above the surface of Mars. There was an error in the spacecraft's trajectory that made it crash into Mars instead of going into orbit. The error was caused by the failed translation of English units into metric units in a segment of ground-based, navigation-related mission software. Later it was found that by comparing Doppler and range solutions with those computed using only Doppler or range data a discrepancy could have been detected in time for corrective action [10].

In the case of Mars Polar Lander, it is hypothesized that a faulty sensor made it turn off its landing thrusters before touchdown. The sensor was triggered by the legs unfolding instead of waiting until they actually touched down on the surface of Mars. Since the engine turned off too soon, the spacecraft fell to the surface at about 50 miles per hour, and crash-landed [11].

A number of the future space exploration missions include rovers. Detecting faults in the rover domain is an even more complex problem. This is because the environmental interactions are unpredictable and ill modeled, and sensors tend to be very noisy.

A perfect example of this is the *Dante II* robot [1]. In 1994, Dante II was deployed in a remote Alaskan volcano to demonstrate remote robotic exploration. While ascending out of the crater, it encountered steep slope and cross-slope conditions that changed the system dynamics. Failure to account for this resulted in the robot falling on its side. *Dante II* was unable to self-right and had to be rescued by helicopter.

All these examples show that faults manifest themselves in subtle ways that can only be detected by continuously monitoring the dynamics of the system. They also show that the system dynamics tend to be different in different operating conditions. For example, for a rover, high power draw on flat ground may be a cause for concern, but on a slope that might be perfectly acceptable.

The general problem is estimating unobservable discrete modes of a system from noisy measurements of continuous variables. In the case of a rover, the continuous variables correspond to the estimated continuous state based on the available sensor data, such as temperature, speed and motor current. The discrete modes correspond to rover states such as "stuck wheel", "broken gear", and "normal operation".

Typically, rovers have limited power and computational resources. In addition, the measurements of continuous variables are noisy and influenced by the external environment. Thus, we need an algorithm for estimating the discrete modes in *real time* from *noisy* measurements of *continuous* variables. We must also take into account the fact that faults are usually *very low probability events*.

In this paper, we investigate how faults and special conditions can be identified autonomously and robustly. Experiments with a rover simulator provide evidence that this approach works.

Our approach is based on a probabilistic technique known as *particle filtering* [2, 6, 8, 12], also called the *condensation algorithm* [4] and *Markov Chain Monte Carlo* (MCMC) [3, 14]. In particle filtering, weighted samples (called *particles*) are used to nonparametrically approximate distributions. The next section describes this in greater detail. Particle filters provide a computationally tractable approach to estimating the state of such hybrid systems.

The remainder of the paper is in the following format:

1. General discussion of various approaches.
2. A review of particle filters.
3. A generalization of particle filters for high cost/low probability events motivated by decision theory.
4. A description of the rover we will simulate.
5. Experimental results from the simulation.

2 Approaches to State Estimation

We formalize the problem by viewing the state of the system as a vector consisting of a discrete component corresponding to the fault and operational modes, and a continuous component corresponding to the continuous state of the system. Since we have a discrete component as well as a continuous one, we cannot apply traditional state estimation techniques, such as Kalman Filtering [5]. This is because Kalman filters track continuous state under the assumption of a unimodal distribution. A common way of tracking a system with discrete and continuous components is to use a bank of Kalman filters [5, 13]. But since failures may occur in any combination, the number of Kalman filters may grow exponentially.

Rich Washington [15] developed an extension that used a POMDP to represent the discrete states of the system. The dynamics within each discrete state are represented using a Kalman filter. This work is a clever way of representing hybrid state. But each Kalman filter can only represent a unimodal posterior. Capturing the full posterior is intractable since there are potentially an exponential number of hypothesis to track. The method therefore tracks only a finite number of most likely hypotheses.

Lerner and others [7] represent the posterior with a mixture of gaussians, where each gaussian represents a hypothesis being tracked. Instead of selecting a subset of hypotheses at each time step, they collapse similar gaussians. This method provides a good approximation of the posterior. It makes the assumption that it is feasible to enumerate all the possible hypotheses.

Williams and Nayak [16] use a qualitative model based technique for fault diagnosis. This approach was successfully used in the spacecraft domain, but it has turned out to be unsuitable for the rover domain because it relies on the system transitioning occasionally from one steady state to another. Thus, it cannot account for the frequent transitions of a rover that are caused by the continuous interactions with the environment.

Shiela McIlraith [9] was one of the first to propose a Bayesian tracking approach for tracking hybrid systems. This approach uses factored sampling techniques, but does not track low probability events well.

3 Particle Filtering

Filtering is defined as the problem of estimating the state of a dynamic system from sensor measurements. As the state evolves, the system receives a sequence of sensor measurements $o_1, o_2, \ldots, o_t$ where o_i represents the observation at time step i. Control inputs $a_1, a_2, \ldots, a_t$ act to alter the state evolution. Filtering estimates the state of the system, s_t, at time t as the posterior distribution:

$$\mathrm{p}(s_t \mid o_t, a_t, o_{t-1}, a_{t-1}, \ldots o_1, a_1) \qquad (1)$$

This is also known as the time t *belief state*, Bel_t. In our approach, the state, $s_t = (c_t, d_t)$, consists of two components: the continuous state variables, c_t, and the discrete fault modes, d_t.

Bayesian Filtering simplifies the filtering problem by assuming that the system state evolves in a *Markovian* way. A Markovian system is one in which past and future states are conditionally independent given the current state. The Markovian assumption will allow us to estimate the state recursively. The initial belief is initialized based on prior knowledge. Often, a uniform prior is chosen for simplicity. In addition, we will make the Hidden

Markov Model assumption that our observations are given by a probability distribution dependent only on the current state.

We are trying to estimate equation (1). s_t is a vector of continuous and discrete variables. Applying Bayes rule, we get:

$$\begin{aligned} Bel_t &= \mathrm{p}(s_t \mid o_t, a_t, o_{t-1}, a_{t-1}, \ldots, a_1) \\ &= \eta \mathrm{p}(o_t \mid s_t, a_t, o_{t-1}, \ldots, a_1) \\ &\qquad \times \mathrm{p}(s_t \mid a_t, o_{t-1}, \ldots, a_1) \end{aligned} \tag{2}$$

where, η is a normalizing constant independent of the state.

Since the observations are given under the Hidden Markov Model assumption this implies that:

$$\mathrm{p}(o_t \mid s_t, a_t, o_{t-1}, \ldots, a_1) = \mathrm{p}(o_t \mid s_t) \tag{3}$$

Substituting (3) in (2):

$$\begin{aligned} Bel_t &= \eta \mathrm{p}(o_t \mid s_t)\mathrm{p}(s_t \mid a_t, o_{t-1}, \ldots, a_1) \\ &= \eta \mathrm{p}(o_t \mid s_t) \\ &\times \sum_{d_{t-1}} \int \mathrm{p}(s_t \mid d_{t-1}, c_{t-1}, a_t, o_{t-1}, \ldots, a_1) \\ &\times \mathrm{p}(d_{t-1}, c_{t-1} \mid a_t, o_{t-1}, \ldots, a_1) dc_{t-1} \end{aligned} \tag{4}$$

Here, we have broken the state s_{t-1} down into continuous, c_{t-1}, and discrete, d_{t-1}, components for explicitness.

Using the Markovian property, (4) may be simplified to:

$$\begin{aligned} & Bel_t = \eta \mathrm{p}(o_t \mid s_t) \\ & \times \sum_{d_{t-1}} \int \mathrm{p}(s_t \mid d_{t-1}, c_{t-1}, a_{t-1}) \\ & \times \mathrm{p}(d_{t-1} c_{t-1} \mid o_{t-1}, \ldots, a_1) dc_{t-1} \\ & = \eta \mathrm{p}(o_t \mid s_t) \\ & \times \sum_{d_{t-1}} \int \mathrm{p}(s_t \mid d_{t-1}, c_{t-1}, a_{t-1}) Bel_{t-1} dc_{t-1} \end{aligned} \tag{5}$$

To implement the recursive Bayesian filter in equation (5), the probability distributions required are:

1. state transition $\mathrm{p}(s_t \mid s_{t-1}, a_{t-1})$
2. observation $\mathrm{p}(o_t \mid s_t)$
3. initial belief Bel_0.

Since there are a large number of components that can fail in any combination at any time, there are potentially an exponential number of discrete states. Tracking the full posterior is intractable for any feasible system. Our approximation uses a technique called *particle filtering* where a set of weighted samples called *particles*, $P_t = (s_{ti}, w_{ti})_{1 \le i \le N}$, approximates the belief Bel_t.

This approach has several benefits. It is non-parametric and can represent a wide range of distributions. Both discrete and continuous variables can be represented with a single particle filter. Particle filters are easily implemented since there is no need to specify a posterior based on the full prior. Instead a posterior distribution needs to be specified for a finite state represented by a sample. A forward simulation of rover kinematics is sufficient for this purpose. Since the accuracy of results can be traded for computational efficiency, particle filters are any-time algorithms. One of the reasons for the efficiency is that in many cases the computational complexity does not heavily degrade as the dimension of the state space increases.

Simple particle filters [3] estimate the probability distribution Bel_t based on the data, which consists of measurements and actions taken by the system. We can think of each particle s_{ti} as a hypothesis about the current state of the system. Starting from an initial particle set P_0, at each step the next particle set P_t is obtained from the previous one P_{t-1} by a recursive update. This is done by sampling P_t from the state transition probability. In effect we are sampling from the distribution,

$$Q_t = \mathrm{p}(s_{ti} \mid s_{t-1\,i}, a_{t-1}) Bel_{t-1}$$

This distribution is also known as the *proposal distribution*. The proposal distribution is not exactly the distribution from equation (5) that we are trying to estimate. To take into account the fact that we have sampled from the wrong distribution, the samples are likelihood weighted by the ratio of the desired distribution to the proposal distribution:

$$\begin{aligned} & \frac{\eta \mathrm{p}(o_t \mid s_{ti})\, \mathrm{p}(s_{ti} \mid s_{t-1\,i}, a_{t-1})\, Bel_{t-1\,i}}{\mathrm{p}(s_{ti} \mid s_{t-1\,i}, a_{t-1}) Bel_{t-1\,i}} \\ & = \eta \mathrm{p}(o_t \mid s_{ti}) \\ & = w_{ti} \end{aligned} \tag{6}$$

This gives an updated set of N weighted samples. We can now transform this into an unweighted set of samples by resampling according to the weights.

The recursive update takes into account the actions taken and maintains the particles according to the distribution of our Bayesian belief state. By design, a simple particle filter tracks well the most likely state of the system. This means that if we use a particle filter directly for fault identification, it will track well the hypotheses representing a *nominal state* of the system and poorly detect the unlikely fault states. This drawback is further complicated by the fact that the size of the discrete component of the state space is exponential in the number of subsystems that can fail, since every possible combination of failures of subsystems could occur. Thus, when applied to a complex system with many failure modes an algorithm based on particle filters could experience *sample impoverishment*. In other words, the particle set P_t would approximate the probability distribution of the belief state ineffectively because most particles would represent similar states and no particles would represent unlikely (though important) states. This motivates an improvement to particle filters which we discuss next.

4 Decision Theoretic Particle Filters

We want a system which will allow us to detect the important states. In order to find such a system we can look to the principles of decision theory. In decision theory, for every state and every action (such as the action to recover from a fault state), there is some expected future loss $l(S, A)$. The goal then is to choose the action which minimizes future expected loss. Given a distribution over states, the expected future loss for an action, A is:

$$r(A) = \int_S l(S, A)p(S)dS$$

With standard particle filters, we can attempt to estimate $r(A)$ using Monte-Carlo integration with particles. However, Monte Carlo integration may converge only very slowly because low probability events might have a very high loss associated with them.

Can we achieve quicker convergence for an estimate of $r(A)$? The answer, somewhat surprisingly, is "yes". In general, for a Monte-Carlo integration of two functions:

$$\int_S f(S)g(S)ds$$

quicker convergence can be found by drawing samples from a distribution proportional to $f(S)g(S)$. In fact, the constant of proportionality *is* the result of the integration.

To apply this approach efficiently, we will only be able to keep around one sample set rather then a sample set for every possible action. We can describe the relative importance of states using a utility function, $u(S)$, which might be derived from a decision theoretic loss function, $l(S, A)$ in several ways. Now, if we keep our samples according to a distribution proportional to $u(S)p(S)$, we expect quicker convergence for monte-carlo integrations of the risk of reporting or not reporting faults.

How do we keep a sample set with a distribution proportional to $u(S)p(S)$? We seek a simple recursive rule which will allow us to maintain particles according to this distribution. First, assume that at time $t-1$, we have particles distributed according to $u(s_{t-1})p(s_{t-1})$. In general, we already have a state transition rule, $p(s_t|s_{t-1}, a_{t-1})$ which will allow us to generate new states at time t given the state at time $t - 1$. Using samples drawn according to $u(s_{t-1})p(s_{t-1})$, we can generate states according to:

$$u(s_{t-1})p(s_t)$$

We can then easily adjust this distribution by reweighting our samples with the ratio:

$$\frac{u(s_t)}{u(s_{t-1})}$$

A resampling of the distribution will now generate samples according to:

$$u(s_t)p(s_t)$$

as desired. In fact, the sample/reweight/resample can all be done in one step (with lower variance) by drawing from a "transition probability" which has the reweighting built into it:

$$\frac{u(s_t)}{u(s_{t-1})}p(s_t \mid s_{t-1})$$

Integration of observations is unaffected by this approach and is done in the same manner as for standard particle filters.

Theoretically, we should alter our initial distribution of samples so as to mimic a distribution proportional to our prior times $u(S)$. However, it is sometimes more convenient to pretend that our prior is uniform in $u(S)p(S)$.

Experiments in (section 6) show that the efficiency of fault detection is significantly improved by this approach.

5 Example Domain

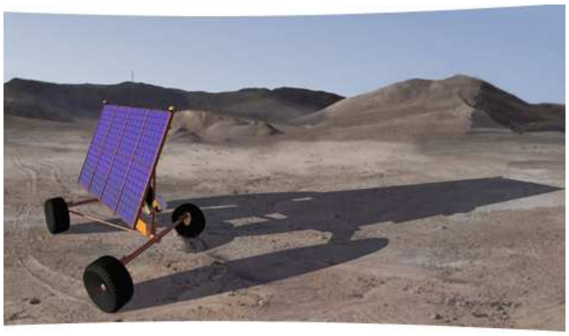

Figure 1: The Hyperion Robot.

The example domain used in this paper is the Hyperion robot (Fig. 1). Hyperion is a mobile robot being developed at Carnegie Mellon University. It will demonstrate autonomous solar powered navigation on Devon Island in the Arctic, in July 2001.

Fig. 2 shows a top view of the robot. Hyperion has four actuated wheels. The back frame is rigid and the front is steerable, but not actuated. $W1$, $W2$, $W3$ and $W4$ are the four wheels and $v1$, $v2$, $v3$ and $v4$ are their respective translational velocities. α is the steering angle and θ, which is not shown in the diagram is the rotation of the frame, i.e. rotation with respect to world coordinates. R_x and R_y represent x and y coordinates in the rover frame and s_x and s_y represent the x and y coordinates in the steering frame. L is the wheel base or rover length and B is the track or rover width.

For the purpose of this paper, we are looking at a subset of possible failures. We are interested in determining if any of the four wheel motors is faulty, any wheel is stuck and locked, or if any of the drive gears are broken. This is important because in the event of any of the above faults, the commanded velocity at one or more wheels may have to be altered to achieve the desired motion. The problem is to determine if any one of these faults occurs given the commanded velocity at each wheel, noisy measurements of the position and orientation of the rover, and the steering angle reported by the steering potentiometer. For the purpose of this paper, we use a simple linearized kinematic model of Hyperion. This is reasonable since sensor measurements come in at $5Hz$ and the motion of the rover is almost linear between consecutive sensor measurements. The wheel positions, P_1, P_2, P_3 and P_4 of each wheel in the rover coordinate frame are as follows:

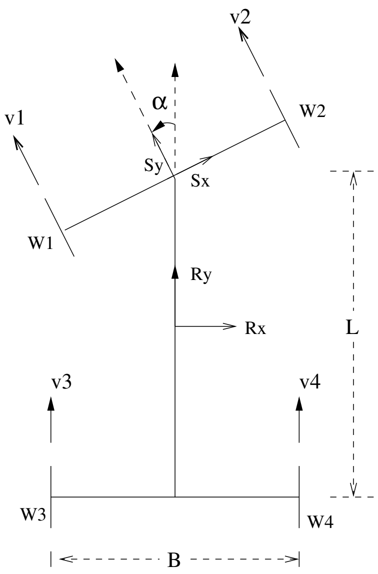

Figure 2: Top view of Hyperion.

$$P_1 = \left(\frac{B}{2}\cos\alpha, \frac{L}{2} - \frac{B}{2}\sin\alpha\right)$$
$$P_2 = \left(\frac{B}{2}\cos\alpha, \frac{L}{2} + \frac{B}{2}\sin\alpha\right)$$
$$P_3 = \left(-\frac{B}{2}, -\frac{L}{2}\right)$$
$$P_4 = \left(\frac{B}{2}, -\frac{L}{2}\right)$$

The instantaneous motion of the robot is represent by the vector $\vec{M} = [\Delta x, \Delta y, \Delta\theta, \Delta\alpha]^{\mathrm{T}}$. The kinematic equations for the instantaneous motion at each wheel are:

$$\begin{aligned}
\vec{M}_1 &= \left[\Delta x + \left(\frac{L}{2} + \frac{B}{2}\sin\alpha\right)\Delta\theta + \frac{B}{2}\Delta\alpha\right]\hat{x} + \\
&\quad \left[\Delta y - \frac{B}{2}\cos\alpha\Delta\theta - \frac{B}{2}\cos\alpha\Delta\alpha\right]\hat{y} \\
\vec{M}_2 &= \left[\Delta x - \left(\frac{L}{2} - \frac{B}{2}\sin\alpha\right)\Delta\theta + \frac{B}{2}\Delta\alpha\right]\hat{x} + \\
&\quad \left[\Delta y + \frac{B}{2}\cos\alpha\Delta\theta + \frac{B}{2}\cos\alpha\Delta\alpha\right]\hat{y} \\
\vec{M}_3 &= \left[\Delta x + \frac{L}{2}\Delta\theta\right]\hat{x} + \left[\Delta y - \frac{B}{2}\Delta\theta\right]\hat{y} \\
\vec{M}_4 &= \left[\Delta x + \frac{L}{2}\Delta\theta\right]\hat{x} + \left[\Delta y + \frac{B}{2}\Delta\theta\right]\hat{y}
\end{aligned} \tag{7}$$

The rover simulator used in the paper minimizes wheel slip for any given set of commanded velocities.

6 Experiment

The ten discrete states that we estimate are – *wheel1 or wheel2 motor or gear broken, wheel3 broken, wheel4 broken, wheel1 locked stuck, wheel2 locked stuck, wheel3 locked stuck, wheel4 locked stuck, wheel3 gear broken, wheel4 gear broken.* The reason that *wheel1 or wheel2 motor or gear broken* are lumped together as a single fault is because when either of the front wheels cannot be controlled, there is no way to control the heading of the rover and the mission will have to be aborted. Aggregating faults that result in the same response reduces the dimensionality of the state space. On the other hand if a motor on one of the back wheels is determined to be broken, then the control input may be modified to produce the desired motion.

The dynamics of the rover are different in each of these discrete states. The dynamics within each discrete state are represented with a set of equations. The continuous variables used to track system dynamics are represented by the vector $\hat{C} = [x, y, \theta, \alpha]^{\mathrm{T}}$ and control by $A = [u_1, u_2, u_3, u_4]^{\mathrm{T}}$, where the $u_i s$ are commanded wheel velocities.

The system belief state is represented with a set of N weighted samples, $P_t = (s_{ti}, w_{ti})_{1 \le i \le N}$. Here each sample is in any one of just 10 discrete states (no multiple failures) and has a value for each particle $s_i = \left[d_i, c_i\right]^{\mathrm{T}}$, where
$d_i \in$ {*normal, wheel*1 *or wheel*2 *motor or gear broken, wheel*3 *broken, wheel*4 *broken, wheel*1 *locked stuck, wheel*2 *locked stuck, wheel*3 *locked stuck, wheel*4 *locked stuck, wheel*3 *gear broken, wheel*4 *gear broken* } . The initial discrete state is assumed to be the normal operation state and each sample has weight $\frac{1}{N}$.

At each time step, the sample s_{ti} is updated to get sample $s_{t+1\,i}$. We first do the discrete state transition by drawing a sample from the discrete state transition function:

$$d_{t+1\,i} \sim \mathrm{p}(d_{t+1\,i} \mid d_{ti}, c_{ti}, a_t)$$

The dynamics represented by the discrete state at $t+1$ are used to estimate the continuous state $c_{t+1\,i}$ by drawing a sample according to:

$$c_{t+1\,i} \sim \mathrm{p}(c_{t+1\,i} \mid d_{t+1\,i}, d_{ti}, c_{ti}, a_t)$$

.

The samples are then weighted by the likelihood of the sample given the sensor measurement. The sensor measurement is the rover pose in (x, y, θ) from GPS and the steering angle α from the steering potentiometer. The measurement matrix is $\vec{O} = [x, y, \theta, \alpha]$.

These N samples are normalized so that the weights sum to 1 and the process recurses.

For the experiment, first the rover is driven with a variety of different control inputs in the *normal* operation mode. At the $17th$ timestep, *wheel3 is stuck* and locked against a rock. *Wheel3* is then driven in the *backward* direction, which causes it to get unstuck and the rover returns to the *normal* operation mode. It continues to operate normally until the *gear on wheel4 breaks* at the $30th$ timestep. This fault is not recoverable and the controller just alters its input based on this state. In this experiment we do not look at multiple simultaneous failures, but they can easily be included by extending the model. Fig. 3 shows the commanded velocity at each wheel and Fig. 4 shows the rover position. For clarity this figure shows the pose at only a select few timesteps. Fig. 5 shows the results obtained by using a simple particle filter, with hybrid state, for this experiment. Even for a simple experiment with 10 discrete states and 4 continuous variables, a large number of samples were required before it was possible

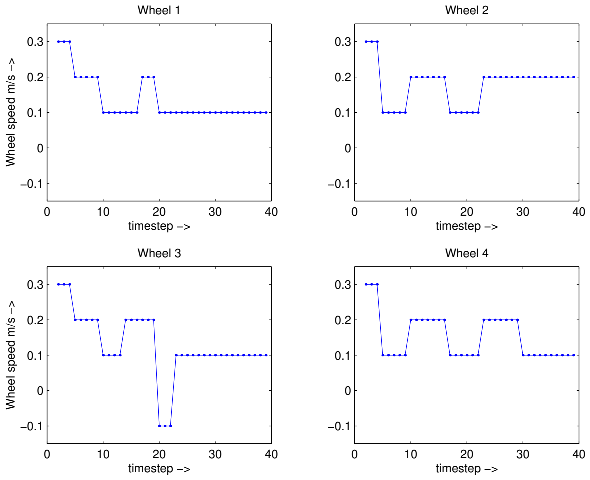

Figure 3: Commanded velocities at each wheel.

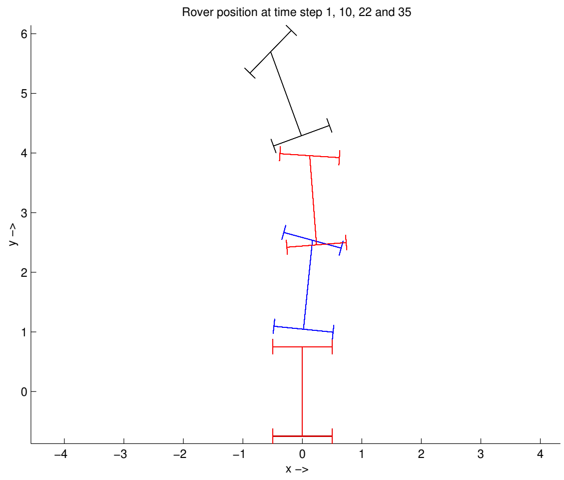

Figure 4: Rover position at time step 1, 10, 22 and 35.

to track fault 10. The reason is that faults are low probability events and when the state is approximated using a small sample set, none of the samples transition to the true state. This simple particle filter does not scale well. For the purpose of fault detection, the low probability fault states are very important to track. We handle this by forward sampling based on the utility *and* probability of the next state transition rather than just the *probability* of the next state transition. Fig. 6 shows the results with this method. The figure shows the most likely state estimate, the sample variance of the filter, and the error in the estimate over 100 repetitions of the experiment. It also shows the variance of this error over a 100 repetitions of the experiment. With 100 samples the filter was able to track the fault states reasonably well and with a 1000 samples

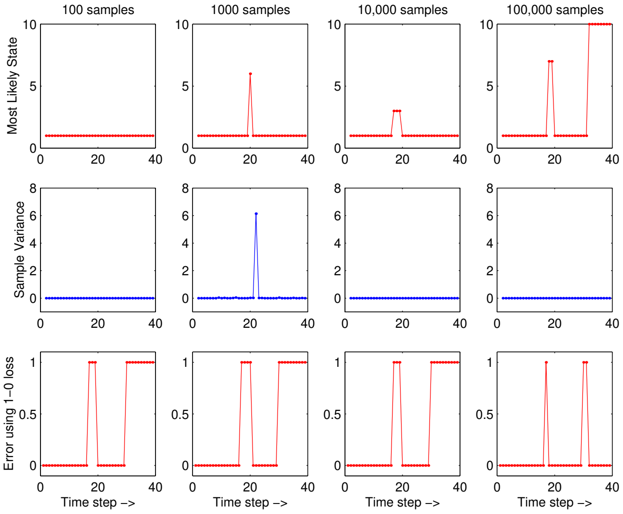

Figure 5: Results with a simple particle filter. Here *(1)normal, (2)wheel*1 *or wheel*2 *motor or gear broken, (3)wheel*3 *broken, (4)wheel*4 *broken, (5)wheel*1 *locked stuck, (6)wheel*2 *locked stuck, (7)wheel*3 *locked stuck, (8)wheel*4 *locked stuck, (9)wheel*3 *gear broken, (10)wheel*4 *gear broken*

.

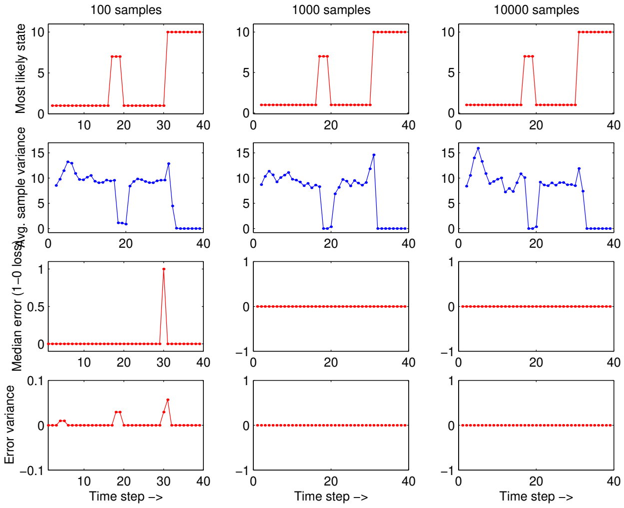

Figure 6: Results with a utility based particle filter

it erred only once.

7 Conclusion

The algorithm proposed succeeds at non-parametric fault identification in our simple domain using only limited real-time computation. The most important step for future work is scaling this approach up to larger domains, showing that it can handle multiple faults and doing (inherently expensive) validation tests on real robots. We anticipate that it will work well in many other fault identification domains and the decision theoretic extension to particle filters may be of interest beyond the fault identification problem.

We are currently working on implementation and testing with a real robot.

References

[1] John E. Bares and David S. Wettergreen. Dante ii: Technical description, results and lessons learned. volume 18 of *International Journal of Robotics Research*, pages 621–64, July, 1999.

[2] A. Doucent. On sequential simulation-based methods for Bayesian filtering. Technical report, Cambridge University, 1998.

[3] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. In *Journal of Artificial Intelligence,*, volume 11, pages 391–427, 1999.

[4] M. Isard and A. Blake. Condensation: conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[5] R. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, (82):34–45, 1960.

[6] K. Kanazawa, D. Koller, and S.J. Russel. Stochastic simulation algorithms for dynamic probabilistic networks. In *Proceedings of UAI*, 1995.

[7] Uri Lerner, Ronald Parr, Daphne Koller, and Gautam Biswas. Bayesian fault detection and diagnosis in dynamic systems. In *Proceedings of the 17th National Conference on Artificial Intelligence*. AAAI, 2000.

[8] J. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. In *Journal of the American Statistical Association*, volume 93, 1998.

[9] Shiela McIlraith. Diagnosing hybrid systems: A bayesian model selection approach. Proceedings of the Eleventh International Workshop on Principles of Diagnosis (DX'00), pages 140–146, June 2000.

[10] NASA. Mars climate orbiter mishap investigation board phase i report. Report published by investigation commmittee., Nov 1999.

[11] NASA. Mars program independent assessment team summary report. Report published by investivation team., March 14 2000.

[12] M. Pitt and N. Shephard. Fitering via simulation: auxiliary particle filter. In *Journal of the American Statistical Association*, volume 94, June 1999.

[13] S.I. Roumeliotis, G.S. Sukhatme, and G.A. Bekey. Fault detection and identification in a mobile robot using multiple-model estimation. Proceedings of the 1998 IEEE International Conference on Robotics and Automation, pages 2223–2228, May 1998.

[14] Sebastian Thrun, Dieter Fox, and Wolfram Burgard. Monte carlo localization with mixture proposal distribution. In *Proceedings of the AAAI National Conference on Artificial Intelligence*. AAAI, 2000.

[15] Rich Washington. On-board real-time state and fault identification for rovers. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, 2000.

[16] Brian C. Williams and P. Pandurang Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of AAAI*, 1996.