Particle Filters for Real-Time Fault Detection in Planetary Rovers

Richard Dearden and Dan Clancy

Research Institute for Advanced Computer Science / NASA Ames Research Center
Mail Stop 269-3
Moffett Field, CA 94035 USA
Email: {dearden,clancy}@ptolemy.arc.nasa.gov

ABSTRACT

Planetary rovers provide a considerable challenge for robotic systems in that they must operate for long periods autonomously, or with relatively little intervention. To achieve this, they need to have on-board fault detection and diagnosis capabilities in order to determine the actual state of the vehicle, and decide what actions are safe to perform. Traditional model-based diagnosis techniques are not suitable for rovers due to the tight coupling between the vehicle's performance and its environment. Hybrid diagnosis using particle filters is presented as an alternative, and its strengths and weaknesses are examined. We also present some extensions to particle filters that are designed to make them more suitable for use in diagnosis problems.

1 INTRODUCTION

Planetary rovers provide a considerable challenge for robotic systems in that they must operate for long periods autonomously, or with relatively little intervention. To achieve this, they need to have on-board fault detection and diagnosis capabilities in order to determine the actual state of the vehicle, and decide what actions are safe to perform.

The diagnosis problem is to determine the current state of a system given a stream of observations of that system. In traditional *model-based diagnosis* systems such as the Livingston system [1] that was a part of the Remote Agent experiment [2,3] diagnosis is performed by maintaining a set of candidate hypotheses (in the Remote Agent experiment, only a single hypothesis was kept) about the current state of the system, and using the model to predict the expected future state of the system given each candidate. The predicted states are then compared with the observations of what actually occurred. If the observations are consistent with a particular state that is predicted, that state is kept as a candidate hypothesis. If they are inconsistent, the candidate is discarded. Traditional diagnosis systems typically use a logic-based representation, and use *monitors* to translate continuous-valued sensor readings into discrete-valued variables. The system can then reason about the discrete variables, and compare them with the predictions of the model using constraint propagation techniques.

Unlike spacecraft, rover performance depends significantly on environmental interactions. The on-board sensors provide streams of continuous valued data that varies due to noise, but also due to the interaction between the rover and its environment. For example, a rover may have a sensor that reports the current drawn by a wheel. In normal operation, this quantity may vary considerably, increasing when the vehicle is climbing a hill, and decreasing on downward slopes. The diagnosis system needs to be able to distinguish a change in the current drawn due to the terrain being traversed from a change due to a fault in the wheel. A second issue for rovers is that their weight and power is very tightly constrained. For this reason, any on-board diagnosis system must be computationally efficient, and should be able to adapt to variations in processor availability. Ideally, we would also like it to adapt based on its own performance, spending more time on diagnosis when a fault is likely to have occurred, and less time when the system appears to be operating normally.

A rover's close coupling with its environment poses a considerable problem for traditional model-based diagnosis systems. A particular sensor reading may be normal under certain environmental conditions, but indicative of a fault in others, so any monitor that translates the sensor reading into a discrete value such as "nominal", or "off-nominal high" must be sophisticated enough to take all the environmental conditions into account. This can mean that the diagnosis problem is effectively passed off to the monitors—the diagnosis system is very simple, but relies on discrete sensor values from extremely complex monitors that "diagnose" the interaction between the system and its environment as part of translating continuous sensor values into discrete variables. To overcome this problem, we need to reason directly with the continuous values we receive from sensors. That is, our model needs to be a hybrid system, consisting of a set of discrete *modes* that the system can be in, along with a set of continuous state variables. The dynamics of the system is described in terms of a set of equations governing the evolution of the state variables, and these equations will be different in different modes. In addition, a transition function describes how the system moves from one mode to another, and an observation function defines the likelihood of an observation given the mode and the values of the system variables.

This hybrid model is very similar to a *partially observable Markov decision process* (POMDP). POMDPs are frequently used as a representation for decision-theoretic planning problems, where the task is to determine the best action to perform given the current estimate of the actual state of the system. This estimate, referred to as the *belief state*, is exactly what we would like to determine in the diagnosis problem, and the problem of keeping the belief state updated is well understood in the decision theory literature. The belief state is a probability distribution over the system states, and is updated by "pushing it through" the system model to produce a new probability distribution over the possible future state of the system, and then conditioning that distribution on the actual sensor values that were observed to remove any states inconsistent with the observations, reduce the probability of any state from which the observations are unlikely, and increase the probability of states that predict the observations well. We will take essentially this approach to the diagnosis problem in this paper.

Unfortunately, the algorithm we described above for maintaining an accurate belief state is computationally intractable for the types of problem we are interested in. Since our model contains both discrete and continuous variables, the belief state is a set of multidimensional probability distributions over the continuous state variables, with one such distribution for each mode of the system. This distribution may not even be unimodal, so just representing it is a complex problem, but applying the model to it to predict future state is infeasible due to the severe computational restrictions on-board a rover. Therefore, an approximation needs to be made. The approach we will take is to use *particle filtering* [4,5].

A particle filter represents a probability distribution using a set of discrete samples, referred to as *particles*, each of which has an associated weight. The set of weighted particles constitutes an approximation to the belief state, and has the advantage over many approximation methods that it can represent arbitrary distributions. To perform diagnosis, we apply the predictive model to each particle individually, and then condition on the observations by multiplying the particle's weight by the probability of observing the sensor readings if the particle represented the true state of the system. To prevent a small number of particles from dominating the probability distribution, the particles are then resampled, with a new set of particles, each of weight one, being constructed by selecting samples randomly based on their weight from the old set. We will discuss particle filters in more detail in Section 3 below.

Particle filters have already proven very successful for a number of tasks, including robot navigation [6]. Unfortunately, they are less well suited to diagnosis tasks. This is because the mode transitions that we are most interested in detecting—namely transitions to fault states—are those with the lowest probability of actually occurring. Thus, there is a risk that there will be no particle in a fault state when a fault occurs, and so the system will be unable to diagnose the fault as no particle will predict the observations well. To apply particle filters successfully to rover diagnosis, we will have to overcome this problem, and we discuss two approaches here. In the first, we increase the number of particles when the current particles seem to be doing a poor job of predicting the observations. The idea is that when the total weight of the particles falls, it indicates that no particle is predicting well. In response to this, the diagnosis system can suggest that a fault may have occurred, and use additional resources (additional particles) to try to identify the fault.

A more promising approach is to think of the particle filter as a convenient way to divide the available computation time between individual states. In this model, the system uses as many particles as computational limitations will allow, and divides them up between the system modes according to how well each mode predicts the sensor readings as before, but also according to how important it is to diagnose each mode. So for example, if a particular mode m predicts the observations well at present, we would ensure that there are always some particles in a fault state m' that is reachable from m with non-zero probability. This means that if the fault did occur, there are guaranteed to be some particles in m' that would predict the observed behaviour after the fault.

In the next section we discuss the rover model in detail. In Section 3 we describe particle filtering and demonstrate its weaknesses when applied to diagnosis problems. This paper describes a work in progress, so in Section 4 we will describe our proposed modifications to the standard particle filter in detail, and in Section 5, present some preliminary results on real rover data, using a simple version of our proposed approach. The final section looks at the relationship between this work and some previous approaches to this problem, and discusses some future directions for this work.

2 MODELLING A PLANETARY ROVER

As we said above, we model a rover as a hybrid system. The discrete component of the rover's state represents the various operational and fault modes of the rover, while the continuous state describes the speed of the wheels, the current being drawn by various subsystems, and so on. Following [7], we model a rover as a tuple $<M, V, T, E, O>$ where the elements of the tuple are as follows:

- M is the set of discrete modes the system can be in. We assume that M is finite, and write m for an individual system mode.
- V is the set of variables describing the continuous state of the system.
- T is a transition function that defines how the system moves from one mode to another over time. We write $\Pr_T(m,m')$ for the probability that the system moves from mode m to mode m'. We may also include a second transition function $\Pr_T(m,a,m')$ which is used when an action a occurs. This gives the probability of moving from m to m' when action a is executed.
- E is a set of equations that describe the evolution of the continuous variables over time. The equations that apply at a given time potentially depend on the system mode, so we write E_m for the equations that apply in mode m. These equations will in general include a noise term to account for random variations in the state variables. Here we will assume Gaussian noise, with the parameters of the Gaussian determined individually for each equation.
- O is a function mapping the system state into observations. We will assume that the observable system characteristics are some subset of the system variables V, with their values corrupted by Gaussian noise (again with parameters that may be a function of the variable, and the system mode), so we write $O(v,m)$ for the observed value of some variable v in mode m.

In addition to these, we will write $\Pr(s'|s)$ for the probability distribution over future states s' given some state s, where s and s' are hybrid states, so $\Pr(s'|s)$ includes both the distribution over the future mode given by $\Pr_T(m'|m)$, and the distributions over the continuous variables given by E_m.

The diagnosis problem now becomes the task of determining the current mode m that the system is in, and the values of all the state variables in V. We will assume for the rest of this paper that all the state variables are observable, that the problem of tracking their values through the noise is relatively straightforward (if the noise in the observations overwhelms the true values of the variables, we have little chance to determine the system mode), and assume that the diagnosis problem is to determine the mode only. Note that the particle filter approach we describe does determine the most likely value for the system variables as well as estimating the mode.

For a system as complex as a rover, we would in general take a component-based approach to modelling, where each component (for example, a wheel) is modelled separately and the complete model is the cross-product of its components. Since each component has much simpler dynamics, and typically a smaller set of discrete modes of operation, this makes the modelling task much easier, but it also simplifies diagnosis by reducing the number of faults that need to be considered to only those in components that aren't behaving nominally. Of course, we can expect a certain amount of dependency in a component-based model (for example, we would expect each wheel to be moving at approximately the same speed in most circumstances). Taking advantage of these dependencies allows us to simplify the diagnosis problem in some cases, for example by identifying a wheel that is faulty because its behaviour doesn't match that of the other wheels. However, we will not discuss these issues in detail here.

The experiments we will present in Sections 3 and 5 use actual telemetry data from NASA Ames' Marsokhod rover. The Marsokhod is a planetary rover built on a Russian chassis that has been used in field tests from 1993–99 in Russia, Hawaii, and the deserts of Arizona and California. The rover has six independently driven wheels, and for the experiments we present here, the right rear wheel had a broken gear, and so rolls passively. The Marsokhod has a number of sensors, including ones that measure body and arm geometry, and battery current. We will restrict our attention to diagnosing the state of the broken wheel, and will therefore use only data from the wheel current and wheel odometry sensors. We will treat each wheel independently in the diagnosis. For each wheel, we have a model, taken from [7], with the following characteristics (in fact, the model of each wheel is identical):

- M consists of 23 system modes, including for example an idle state, the wheel current rising after a command is given, uniform speed driving, and a total of 14 different fault states.
- V consists of variables for the wheel current and wheel speed, and the derivatives of current and speed.
- T is a fairly sparse matrix, with at most six successors for any given mode. The probability of a transition to a fault state is 0.01 or less. We include a separate transition function for the start and end of a command.

1. Create a set of n particles where each particle p has a state p_s and a weight p_w. p_s is sampled randomly from the initial state distribution, and $p_w = 1$.
2. For each time step, do:
   a. For each particle p, do:
      i. Select a future state p_s' for p by sampling from $\Pr(p_s' | p_s)$, the distribution over possible future states given by the model.
      ii. Re-weight p by multiplying its weight by the probability of the observations o given p_s' as follows:
      $$p'_w = \Pr(o \mid p_s) p_w$$
   b. Resample n new particles p_i by copying the current particles where each particle p is added to the new samples with the following probability:
   $$\Pr(p_i = p) = \frac{p'_w}{\sum p'_w}$$

Figure 1: The particle filtering algorithm.

- The state equations in E consist of the previous value plus a constant term and noise. The noise is Gaussian with standard deviation in the range 0.001 to 1.0, and the equations are independent for each state variable.
- The equations in O are independent for each variable (but vary depending on the mode), and consist only of the variable's value plus Gaussian noise. Again, the standard deviation varies from 0.001 to 1.0.

3 PARTICLE FILTERS

A particle filter approximates an unknown probability distribution using a weighted set of samples. Each sample or particle consists of a value for every state variable, so it describes one possible complete state the system might be in. As observations are made, the transition function is applied to each particle individually, moving it stochastically to a new state based on its current dynamics, and then the observations are used to re-weight each particle to reflect the likelihood of the observation given the particle's new state. In this way, particles that predict the observed system performance are highly weighted, indicating that they are in likely states of the system [6]. The major advantage of particle filters is that their computational requirements depend only on the number of particles, not on the size of the system. This is of huge importance to us as it allows us to do diagnosis in an anytime fashion: increasing the number of particles when there is computation time available, or when we suspect a fault has occurred, and decreasing the number when other operations require the processor. To implement a particle filter, we require three things:

- A probability distribution over the initial state of the system.
- A model of the system that can be used to predict, given the current state according to an individual particle, a possible future state of that particle. Since T is stochastic, and E includes noise terms, the predictive model selects a new state for the particle in a Monte Carlo [8] fashion, choosing by sampling from the probability distribution over possible future states.
- A way to compute the likelihood of observing particular sensor values given a state. In our case, this is given by the observation function O.

The particle filtering algorithm is given in Figure 1. Step (i) is the predictive step, where a new state is calculated in a Monte Carlo way for each particle, and this new state is then conditioned on the observations in step (ii). In the case of our rover model, the initial distribution is uniformly in the *Idle* mode, in which there is no power to any of the wheels. The predictive step is performed by applying T to each particle, and then applying the appropriate equations from E to the state variables, sampling values from the Gaussian error terms. Once the particles have been re-weighted, we can then calculate the probability of each mode simply by summing the weights of the particles in the mode.

Problems with particle filters for diagnosis

Unfortunately, there are a number of difficulties in applying particle filters to diagnosis problems. In particular, the filter must have a particle in a particular state before the probability of that state can be evaluated. If a state has no particles in it, the assumption is that its probability of being the true state of the system is zero. This is a problem in diagnosis problems because the transition probabilities to fault states are typically very low, so particles are unlikely to end up in fault states during the Monte Carlo predictive step. Without a particle in a fault state, that state will continue to be given zero probability of having occurred, even if no other state is doing a good job of predicting. This situation is known as *sample impoverishment*.

Figure 2 illustrates this problem. Each graph shows the most likely modes that the wheel is in, shown over part of one of the trials in which the wheel is initially idle, and then at step 12 is commanded to drive forward at a fixed speed. The graphs on the left show the performance of Wheel 1, which is operating nominally. The graphs on the right show the performance of Wheel 6, which is faulty. In the top line, the probability of the fault occurring is 0.1 rather than its true value of 0.01. Here the fault is quickly detected in Wheel 6. The middle line of graphs shows the same situation with the fault probability set to 0.05. Here there is a significant delay between the wheel being commanded, and the detection of the fault. In the bottom line of graphs, the fault probability is set to its true value, and in this case the fault is not successfully detected because insufficient particles enter the fault state. One might expect that once a particle enters the fault state its weight would be high since it would predict well, and at the re-sampling step it should lead to several new particles being created. Unfortunately, this did not occur in this situation because although some particles did enter the fault state, their continuous parameter values did not agree with the observations well, so they still had low weights. The continuous parameters did not match because each of the particles that entered the fault state came from the *CommandedRunning* state, in which the current and wheel speed are expected to be much higher than the observed values. In the next section we examine two techniques for overcoming this problem and ensuring that faults are detected even when their probability of occurring is low.

4 USING DOMAIN KNOWLEDGE TO DIRECT THE PARTICLE FILTER

The simplest solution to the sample impoverishment problem is to increase the number of particles being used. Given the constraints imposed on on-board systems, this approach is probably unrealistic. The data presented above used 10,000 particles per wheel, and runs in Java in approximately 0.5s per update on a 750MHz Pentium 3. This is probably at the upper limit of the number of particles we could expect to use on-board a rover—the time available for diagnosis is longer, but the computation will be much slower. Thus running with ten times as many particles (which is roughly equivalent to multiplying the fault probability by ten) is probably impractical on the rover, and even 10,000 particles may be unrealistic as the model gets more complex. This could be somewhat overcome by only increasing the number of particles when there is some evidence that the system is predicting poorly. In order to achieve this, we need some measure of when this occurs. The obvious measure is to look at the total weight of the particles after conditioning on the observations. If no particles are predicting the observations well the total weight should drop. Unfortunately, in practice this is rarely useful because there are a number of other possible causes for this behaviour. For example, particles moving from a state in which there is high confidence in the sensor readings to a state with more sensor noise will tend to drop in weight even if they are still predicting the observations well. We see this in the Marsokhod model because the *Idle* mode has relatively large variance for the observation noise, whereas the *CommandedRunning* mode has smaller variance, so the total particle weight increases when the system moves from the *Idle* to the *CommandedRunning* mode, even for wheel 6 where *CommandedRunning* predicts the observations poorly.

Another way to reduce the likelihood of sample impoverishment is to bias the distribution of particles in favour of states which correspond to important faults. In this approach, the particle filter no longer approximates the true belief distribution, but instead uses some oracle to identify "important" future states, and allocates some portion of the particles to those states. The particle filter can then be thought of as a way of dynamically allocating our available computation. Each system mode is assigned a fraction of the available computation time, in the form of a number of particles. As in the standard particle filter, particles move from state to state via T and E, and via the resampling step, but the oracle ensures that every state it considers important is assigned at least some small fraction of the available computation. This ensures that if a transition to one of the important states occurs, there will be particles in the state so its probability will be non-zero. If the particles in one of these states predict the observations well (which they should if it is the true state), their weight will then increase, at which point the normal operation of the particle filter increases the number of particles in the state.

The question that remains is how to implement the oracle. For a complex system such as a planetary rover, with many components each with its own set of possible failure modes, there are exponentially many possible failure modes, so this is a non-trivial problem. However, one approach that seems promising is to use a traditional model-based diagnosis system such as Livingston [1]. Livingston operates much more quickly than hybrid approaches because it does not consider the continuous dynamics of the system. Instead, it uses a discrete abstraction of the problem, and tries to identify likely states that are consistent with the observations. We pointed out in the introduction that this approach is not in general suitable for diagnosing rovers, but it is suitable for the purpose of identifying likely system modes to

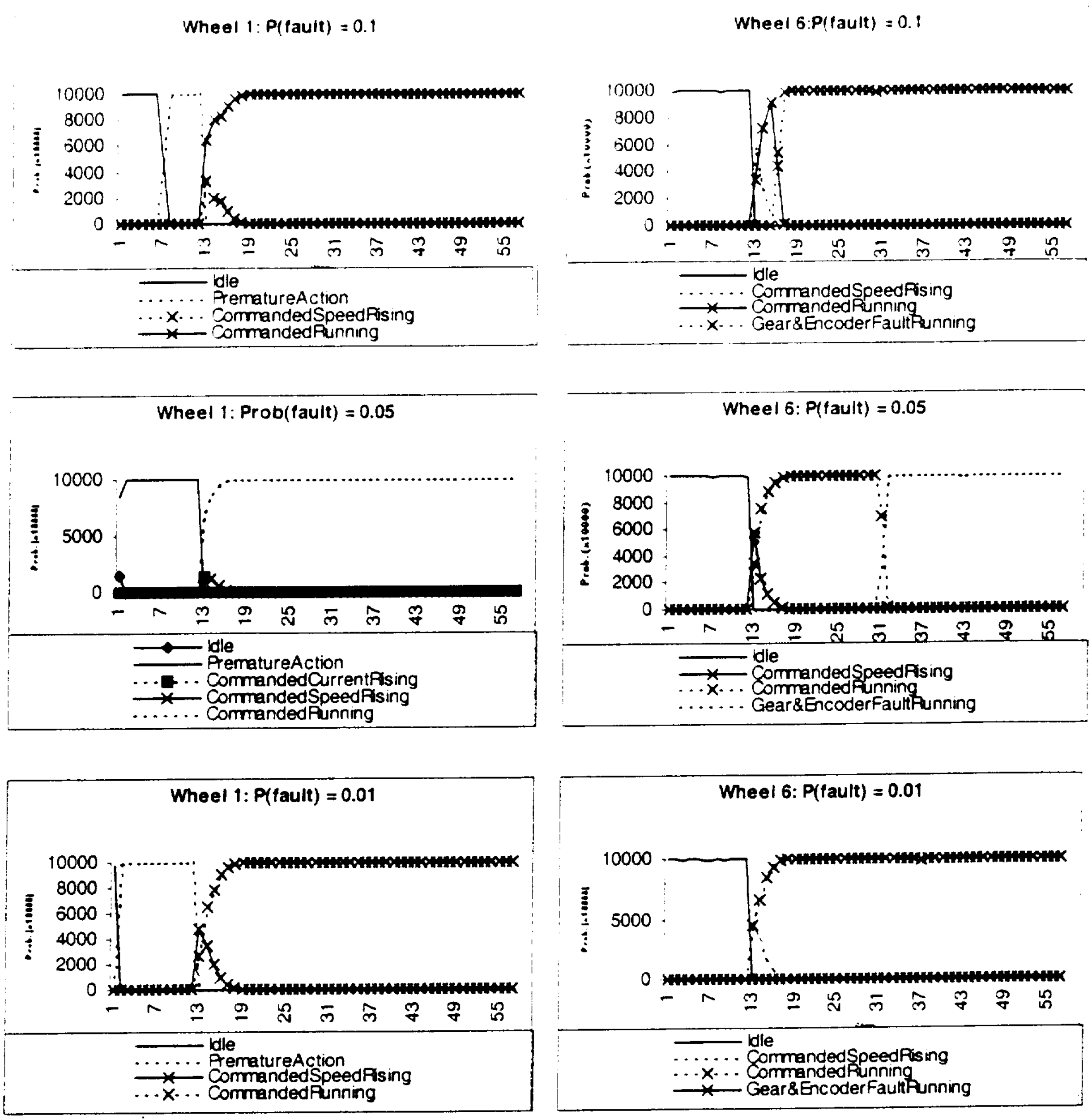

Figure 2: Particle filter results for wheels 1 (nominal performance) and 6 (faulty, with a broken gear). In the top row the probability of the fault is ten times its true value, in the middle row it is 5 times, and the bottom row has the true fault probability. The fault is easily detected in the top row, detected after a delay in the middle row, and not detected in the bottom.

apply some computation in. By asking Livingston to provide a number of candidate hypotheses that explain the observations, we can with high probability be confident that the true mode is in one of the hypotheses. We then assign some computation to all of the modes consistent with the hypotheses. The speed of the qualitative system should mean that using the oracle would not significantly impact computation, which still should depend only on the number of particles. The integration of Livingston with the particle filter approach is currently work in progress, as it adds a number of additional complications including building an additional system model, and ensuring that the discrete and hybrid models agree with one another and can easily be translated back and forth.

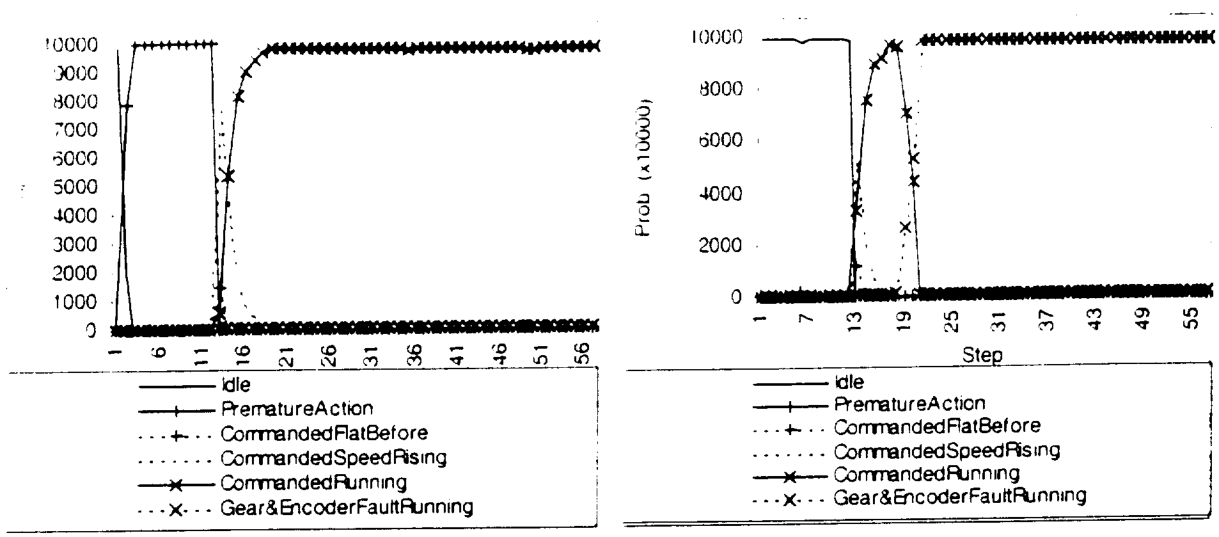

Figure 3: Results for particle filter with bias. All states with > 25% probability were used as starting points for the forward search, and 0.5% of the particles were assigned to each of the found states. On the left are results for wheel 1, and on the right for wheel 6.

For simpler systems such as the Marsokhod wheel diagnosis we have used in this paper, the Livingston-based approach is unnecessary. Instead, we can use an oracle based on forward search from the current high-probability states. Since each system mode in this model has at most six possible successors, and there are typically only two to three high probability modes at any time (often one of these succeeds another), we find in practice that in most cases a simple one-step look-ahead search adds fewer than five modes to those that already contain particles.

5 RESULTS

The results we present here are based on the Marsokhod model we described in Section 2. Dr. Rich Washington supplied the model and the data, which came from his work on using Kalman filters for rover diagnosis [7]. The only changes made to the model were to make it suitable for use with a particle filter; no changes were made to model parameters or transition probabilities. To demonstrate our approach we use a small piece of one of the telemetry data files (the same piece used in Section 3) in which the rover is initially idle, and then a drive command is issued, resulting in an increase in current to each wheel, followed by a corresponding increase in speed, and then a constant speed. As before, wheel 6 is faulty, with a broken gear (this corresponds to the *Gear&EncoderFaultRunning* state in the model).

Figure 3 shows the results for the biased particle filter. For these results we used single step forward search from all states with probability > 0.25 to select the set of bias states. Each of these states was then guaranteed to receive at least 0.5% of the total number of particles at each re-sampling step. The left hand graph is the probable states for Wheel 1, as before. Like the graph in the bottom row of Figure 2, the *PrematureAction* state was given high probability before step 13. This state appears in situations where the effects of an action are seen before the signal to perform the action is seen, due to problems with the rover telemetry. In this case it is a spurious result due to the model of the *Idle* state not allowing sufficient noise in the observations. A small adjustment to the model would remove this problem, which is only present in the data for two of the wheels. The right hand graph shows the same data for Wheel 6. In this case, the fault state is found at step 20, seven steps after the command to drive the wheel was observed. This compares favourably with the results in Figure 2, where the fault was detected three steps after the command in the top row (ten times probability), and 19 steps after it in the middle row (five times probability).

6 DISCUSSION AND RELATION TO OTHER WORK

One closely related piece of work is Verma et al's decision-theoretic particle filter [9]. The problem they are attempting to solve is essentially the same one we are interested in, but their approach is to assign a utility to every state and use the utility to change the distribution they draw the particles from. Effectively, this is equivalent to altering the transition function so that the probability of a transition from state s with utility $u(s)$ to state s' with utility $u(s')$ becomes

Prev State Init. This has a similar effect to increasing the probabilities as we did in Figure 2. For relatively simple diagnosis tasks such as the one we have presented here, the approaches seem very similar. However, for more complex tasks, we believe that using a discrete diagnosis tool as an oracle to direct the computation of the particle filter will allow us to make more effective use of the available computation than the utility-based method, which will increase the probability of transitions to a potentially very high number of fault states, especially as any reasonable utility function would give all fault states a high utility.

Another related effort is the work of Washington [7] that applies Kalman Filters to this problem. In this work, the continuous dynamics in each mode is tracked by a set of Kalman filters. The main problem with this approach is that the number of filters tends to increase over time because each time a transition is made to a state the initial conditions for the filter are different, and filters with different initial conditions cannot be combined. This is not a problem for particle filter-based approaches because the particle filters can represent arbitrary distributions over the parameter values, so particles entering a state with two different sets of initial conditions will form a bi-modal distribution. As we said above, we used the model and data from this paper in our own work. We see fewer errors in the mode identification with our approach than in Washington's paper, although we are sometimes slower to identify the fault, and our computational requirements are somewhat higher.

Lerner et al [10] use linear-Gaussian Bayesian networks to represent a hybrid system, and perform diagnosis by computing the true belief state at every step. This avoids the problems we have described, which are directly caused by the approximation approach. However, the computational requirements are far too high to be used on-board a rover.

As we said in the introduction, this is a work in progress. There is still much work to do on the problem of how to integrate a model from Livingston with this system to act as an oracle. We have demonstrated that a simple look-ahead search performs quite well, but this is clearly inadequate for large diagnosis problems. We are also examining a number of other approaches to improving diagnosis with particle filters, such as backtracking when prediction is poor, and re-sampling past states based on observations that occurred more recently. Finally, we are investigating how a diagnosis system of this type would fit with the CLARAty rover architecture [11] being used for future NASA missions to Mars. We are actively involved in developing these approaches to be integrated with the Mars'07 mission.

We would like to thank Dr. Richard Washington for providing the model of the Marsokhod rover we used for this paper, along with the telemetry files from his field tests with the rover. This work has also been influenced by discussions with Dr Washington, Vandi Verma, and Dr. Robin Morris.

REFERENCES

[1] B. Williams and P. Nayak, "A Model-based Approach to Reactive Self-Configuring Systems" *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference*. AAAI Press, Portland OR, 1996.

[2] N. Muscettola, P. Nayak, B. Pell and B. Williams, "Remote Agent: To Boldly Go Where No AI System Has Gone Before." *Artificial Intelligence 103(1-2)*, August 1998.

[3] D. Bernard, G. Dorais, C. Fry, E. Gamble, B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. Nayak, B. Pell, K. Rajan, N. Rouquette, B. Smith and B. Williams, "Design of the remote agent experiment for spacecraft autonomy." In *Proceedings of the IEEE Aerospace Conference*, March 1998.

[4] A. Doucet, "On Sequential Simulation-Based Methods for Bayesian Filtering." *Technical Report CUED/F-INFENG/TR.310, Department of Engineering, Cambridge University*, 1998.

[5] M. Isard, A. Blake, "CONDENSATION: Conditional Density Propagation for Visual Tracking." *International Journal of Computer Vision*, 1998.

[6] D. Fox, W. Burgard, S. Thrun, "Markov Localization for Mobile Robots in Dynamic Environments." *Journal of Artificial Intelligence, 11*, 1999.

[7] R. Washington, "On-Board Real-Time State and Fault Identification for Rovers." In *Proceedings IEEE International Conference on Robotics and Automation*, April 2000.

[8] W.R. Gilks, S. Richardson and D.J. Spigelhalter, eds. *Markov Chain Monte Carlo in Practice*. CRC Press, 1996.

[9] V. Verma, J. Langford and R. Simmons, "Non-Parametric Fault Identification for Space Rovers." *International Symposium on Artificial Intelligence and Robotics in Space (iSAIRAS)*, June, 2001.

[10] U. Lerner, R. Parr, D. Koller, G. Biswas, "Bayesian Fault Detection and Diagnosis in Dynamic Systems." In *Proceedings 17th National Conference on Artificial Intelligence*, July 2000.

[11] R. Volpe, I.A.D. Nesnas, T. Estlin, D. Mutz, R. Petras, H. Das, "The CLARAty Architecture for Robotic Autonomy." *Proceedings of the 2001 IEEE Aerospace Conference*, Big Sky Montana, March 10-17 2001.