Proceedings of the 2000 IEEE
International Conference on Robotics & Automation
San Francisco, CA • April 2000

On-Board Real-Time State and Fault Identification for Rovers

Richard Washington[†]
Autonomy and Robotics Area
NASA Ames Research Center, MS 269-2
Moffett Field, CA 94035
richw@ptolemy.arc.nasa.gov

Abstract

For extended autonomous operation, rovers must identify potential faults to determine whether its execution needs to be halted or not. At the same time, rovers present particular challenges for state estimation techniques: they are subject to environmental influences that affect sensor readings during normal and anomalous operation, and the sensors fluctuate rapidly both because of noise and because of the dynamics of the rover's interaction with its environment. This paper presents MAKSI, *an on-board method for state estimation and fault diagnosis that is particularly appropriate for rovers. The method is based on a combination of continuous state estimation, using Kalman filters, and discrete state estimation, using a Markov-model representation.*

1 Introduction

Rovers that operate autonomously for extended periods of time must be able to detect and diagnose anomalous situations and recover from faults that do not require ground-operator intervention. Rovers present characteristics that make this problem challenging. They receive streams of continuous-valued sensor data that fluctuate with noise and environmental interactions. From this data they must infer the presence of nominal and off-nominal states, but these states depend on the situation. The boundaries of states can change depending on the context: for example, a high current driving uphill or over an obstacle may be normal, but a high current on smooth, flat ground may indicate an anomaly. In addition, the set of anomalous states can change: for example, the set of possible anomalies will be different for driving and taking pictures. Moreover, rovers are limited in power and weight, which in turn limits processor speed and memory. They must therefore use computationally efficient procedures for state estimation.

The rover state can be thought of as transitioning among a set of possible, qualitatively different states. These states may correspond to operational modes of the rover (e.g., driving) or fault modes (e.g., broken wheel gear)[1]. Transitions may be explicit, based on actions taken by the rover executive (e.g., stopped to driving), or implicit, based on sensor information (e.g., wheel encoder nominal to broken).

Sensor failures may be inferred from the diagnosis, and a failure should then influence future state estimation and diagnosis. For example, driving with a broken encoder will give rise to different "normal" sensor readings for the broken encoder.

Two major branches of work for state estimation and fault diagnosis are Kalman filters from control theory and qualitative model-based diagnosis from artificial intelligence. State estimation for rovers exceeds the capabilities of the current approaches.

Qualitative model-based techniques for diagnosis [2, 6] rely on the system transitioning occasionally from one steady state to another. Rovers receive rapidly-changing streams of continuous-valued sensor data. In addition, the model-based techniques often rely on a snapshot of the system, disregarding history. But in fact the history may be critical to reach a correct diagnosis; the probability of a particular failure may be significantly different based on the prior state. In addition, the qualitative approaches rely on global consistency to compensate for the local inaccuracy of a qualitative model; this can be expensive to compute,

Techniques for state estimation of continuous values, such as Kalman filters [3], can track multiple hypotheses [7, 10], but they lack methods for automatially choosing which states to track. Tracking all possible

[†]NASA contractor with Caelum Research Corporation

[1]In fact, the complete set of states is the cross product of the operational modes with the powerset of the possible fault modes. For modeling reasons some states are often combined with equivalent or similar states.

0-7803-5886-4/00/$10.00© 2000 IEEE

states is infeasible.

In this paper, we present MAKSI (Markov And Kalman State Identification), which combines continuous probabilistic state estimation using Kalman filters (KFs) with discrete qualitative state estimation using a Markov-model representation [4]. The discrete states correspond to qualitatively different modes of the rover (driving nominally, idle, stuck wheel, etc), to each of which is associated a model of operation represented as KF parameters. The development of MAKSI arose from experience with model-based techniques in the NASA Ames 1999 Marskohod field test, and it borrows some ideas from the model-based techniques. Differences include the addition of quantitative information via KFs, context-specific probabilities of state transitions, and computational guarantees.

On-board techniques must work within constraints on computation and memory. MAKSI builds on our own and others' work on efficiently tracking belief states [9, 1] and continuous variables [7].

2 Combining discrete and continuous state estimation

The MAKSI approach to state estimation is built from Kalman filtering and Markov-model representations. Before discussing the complete approach, we briefly introduce the components from which it is built.

2.1 Kalman filtering

The Kalman filter (KF) is designed to estimate the state of a process given observations. Here we consider the standard, discrete KF, which is sufficient for our initial experiments. More complex KFs can be formulated for nonlinear processes and observations and for continuous time. In the standard, discrete KF, the process is assumed to evolve linearly given the previous state and a control input:

$$x_{t+1} = A_t x_t + B u_t + w_t \quad (1)$$

where x_t is the state (vector) at time t, u_t is the control input, and w_t is white, zero-mean, normally-distributed noise. The process is observed through measurements related linearly to the process state:

$$z_t = H_t x_t + v_t \quad (2)$$

where z_t is the measurement (vector) and v_t is white, zero-mean, normally-distributed noise.

At each time step, the estimate of the state is first updated using information about the state model, the control input, and the process noise. This estimate (the *a priori* state estimate, $\hat{x}_t^-$) is then fed through the observation model. The observation model is used to predict what observations should be seen. These are compared against the actual observations, and the difference is used to modify the *a priori* state estimate, arriving at the new, *a posteriori* state estimate, $\hat{x}_t^+$.

The observation noise, in the form of a covariance matrix, is used to update the *Kalman gain matrix*, which is the weighting factor used to combine the *a priori* state estimate and the observation differences. The gain matrix also depends on the state error covariance matrix, which is defined as:

$$P_t = E[(x_t - \hat{x}_t)(x_t - \hat{x}_t)^T] \quad (3)$$

The error covariance matrices are updated over time.

The KF finds the optimal estimate of the process state (under the assumptions of the model), minimizing the expected least-squares error. Briefly, the filter operates by first predicting the state using the state update equations, then correcting the prediction using an observation. Error covariances of the state and the observation contribute to "weight" the model, balancing the state model (the prediction) and the observation. Details of KFs can be found in [5, 3], among others.

2.2 Markov-model representation

While Kalman filters handle the problem of state estimation in a continuous space, they do not offer any assistance when the state branches into two or more qualitatively different states. In contrast, the standard model for Markov decision processes (MDPs) is a set of discrete states with probabilistic transitions among the states. In the case of partially-observable MDPs (POMDPs), the estimation of the current system state is represented by a probability distribution over the set of discrete states.

A standard POMDP model consists of a set of states, S, a set of actions A, and a set of observations O. The model also contains a transition matrix T, of size $|S| \times |S| \times |A|$, where t_{ijk} is the probability of transitioning from state s_i to state s_j when action a_k is chosen. In a standard POMDP model, there is an observation matrix Q of size $|S| \times |O|$, where q_{ij} is the probability of seeing observation o_j when in state s_i. In MAKSI, we do not have this direct information, but we use an indirect calculation via the Kalman state estimate.

Finally, the probability distribution over states at time t is denoted as $\pi(t)$, where $\pi_i(t)$ is the probability that the true state is s_i given information about actions and observations.

Given an observation o_j and an action a_k, the state distribution is updated according to the following formula:

$$\pi_i(t) = \frac{q_{ij} \sum_{1 \le l \le |S|} p_{lik} \pi_l(t)}{\sum_{1 \le m \le |S|} q_{mj} \sum_{1 \le l \le |S|} p_{lmk} \pi_l(t)}, \quad (4)$$

As mentioned earlier, we modify this (see the following subsection) to use an indirect computation of q_{ij}.

Since the goal of state identification is to find the best estimation of the current state as a passive operation, the parts of the POMDP model for control (rewards and policies) are not applicable to this problem.

2.3 Combining discrete and continuous models

The basic idea behind MAKSI is to consider the system as a set of discrete states, but rather than treating each as a static situation, the dynamics within the state are represented using a Kalman filter. We start from the POMDP model and augment it with elements from the KF model. We use the term *discrete state* to represent the qualitatively distinct, POMDP-level discrete state of the system, and the term *system state* to represent the state vector of the actual system parameters at the KF level.

As in the POMDP model, MAKSI represents the system as a set of discrete states S and a set of discrete actions A. We define a set of *transition actions* Θ as:

$$\Theta = \left[\bigcup_{a \in A} \{start(a), end(a)\} \right] \cup \{null\}.$$

These correspond to starting action a, ending action a, and the null action (an implicit, data-driven transition). Similar to the POMDP model, we have a transition matrix T of size $|S| \times |S| \times |\Theta|$, where t_{ijk} is the probability of transitioning from discrete state s_i to state s_j when transition action θ_k is taken[2]. Since there is a difference between transitioning from s_i to s_i and remaining in state s_i (see below), we consider the probability of remaining in state s_i given transition action θ_k to be $1 - \sum_{1 \le j \le |S|} t_{ijk}$.

Each discrete state s has associated with it a KF model and a set of constraints describing the *value space* $V(s)$ of possible system state values. In general the value space can be any subset of the possible system state values; for efficiency, we restrict the constraints to be univariate and linear, defining a (potentially infinite) hypercube value space. The KF model has a state and observation vector of the same length, where state element $\hat{x}[i]$ is the best estimate of the value observed as observation element $o[i]$.

Unlike the standard POMDP model, the observation probability matrix Q cannot be statically predicted from the discrete state. Instead, we approximate this by an indirect computation via the KF state:

$$q_{ij} \approx Prob(o_j|K) \cdot Prob(K|s_i) \quad (5)$$

The element $Prob(o_j|K)$ is itself approximated as the volume of the multidimensional normal distribution function, described by the Kalman estimated state and state error covariance, beyond o_j (more precisely, 1– the volume of the minimum error ellipse enclosing o_j). In the multivariate case this is further approximated as a product over individual dimensions:

$$Prob(o_j|K) \approx \prod_{1 \le i \le n} f(o_j[i], \hat{x}(K)[i], P(K)[i,i]) \quad (6)$$

where n is the length of any observation vector, $o_j[i]$ is the ith element of the observation vector o_j, $\hat{x}(K)$ is the KF estimated state, $P(K)$ is the KF state error covariance matrix, and $f()$ is the probability that an individual observation element is predicted by the state estimate and standard deviation in that dimension[3]. The element $Prob(K|s_i)$ is the volume of the Kalman-described normal distribution function that falls within the value space of s_i; this is approximated for computational efficiency as a product of the individual dimensions:

$$Prob(K|s_i) \approx \prod_{1 \le i \le n} Prob(\hat{x}(K)[i] \in V(s_i)[k]). \quad (7)$$

The normal distribution computations are performed efficiently by table lookup and interpolation.

The approximation to the observation probability q_{ij} is then plugged back into the POMDP state distribution update function (Eq. 4), which is used to arrive at the new probability distribution over possible states.

The update function can be seen intuitively as combining context-specific probability (the transition matrix), data-model compatibility ($Prob(K|s_i)$), and model predictiveness ($Prob(o_j|K)$). So a discrete state with a high probability in the state distribution will be appropriate for the context, its model will produce a state estimate that is highly compatible with the value space constraints, and the observations will be highly consistent with the state estimate.

[2]Note that the implicit transitions are thus dependent on the time step. See Section 4 for possible remedies for this.

[3]This assumes independence of the observation vectors, which is a simplifying, but not necessarily accurate assumption.

One complication of using dynamic state models is that the KF system state depends on the initial conditions of the model. As the system predicts a transition from one discrete state to another, the system state in the new discrete state must inherit the system state from the previous state. Thus not all instances of a discrete state are equivalent. For example, consider two discrete states, one with wheel current rising, a second with steady wheel current. A later transition from rising to steady will imply a higher steady wheel current than an earlier transition. This is an important difference from standard POMDP-model representations and raises the danger of model explosion.

MAKSI limits this by maintaining a constant-size distribution over discrete states. makes Thus the POMDP state distribution update is constant with respect to the number of states [9][4]. The KF updates involve matrix multiplications and inverses, so they are of order $\mathcal{O}(n^3)$, where n is the size of the KF system state vector[5]. If the system state vector can be decomposed into independent subsets, the KF update depends on the size of the largest subset.

The danger of limiting the discrete state distribution to a constant size is that the state distribution update formula may produce a null distribution, with all elements 0. We have shown in [8] methods for overcoming this problem; these will be incorporated into the final implementation.

We thus have a way of trading off computational complexity for model accuracy. The size of the discrete state distribution can be tailored to the computational constraints of the application; in planetary rovers, with their relatively impoverished computational power, this may be essential.

3 Experimental validation

We have constructed a prototype implementation of MAKSI and tested it on telemetry data gathered from the Marsokhod rover. The Marsokhod is a medium-sized planetary rover with six, independently driven wheels. For the experiments, the right rear wheel had a broken gear, so it rolled passively. The Marsokhod is instrumented with sensors that measure pose, wheel odometry and currents, and battery currents.

The data used in this paper were collected in an outdoor "sandbox," which is a gravel and sand area with assorted rocks and small hills. The only actions for this preliminary experiment were driving commands, in arbitrary directions and with varying topography.

For the prototype system, we used wheel current and wheel speed (differences between successive encoder values), along with the variation for current and speed (absolute value of differences between successive values). We ran 6 independent state identification processes, one per wheel. Each state identification process had identical starting conditions, so the only difference was in the data each received.

The discrete states corresponded to idle states, driving states, and intermediate states (ramping up current and speed at the beginning of an action, dropping current and speed at the end of an action). These states were replicated for a small set of fault modes, including a stalled motor, a broken gear, and a broken gear and a broken encoder (the latter of these was in fact the case on the right rear wheel). The KF models were constructed crudely and by hand to model the approximate dynamics in each situation; more careful and detailed models should lead to more accurate state identification, but the hypothesis was that this would be sufficient for the prototype test.

See Figure 1 for data from one such experiment (*Experiment 1*). States 0–2 are non-commanded idle and transition states, states 3–8 are normal driving states (state 6 in particular is steady-state driving), and states 9–22 are error states (19–22 are the broken gear and broken encoder condition). In general the highest-probability state corresponds to the correct state. The broken wheel is correctly diagnosed (because of the lack of current variation); this requires a number of data points to overcome the small a priori probability. However, in the left rear wheel, an interval of flat motor current leads to a momentary shift of belief towards a fault state (a broken wheel). When the data vary again, the belief in the broken wheel disappears.

Although the preliminary results have been encouraging, a handful of cases cause problems. Figure 2 shows one case (*Experiment 2*) where, for an unknown reason, the currents flattened out for long enough that 4 of the 5 working wheels incorrectly identified a fault state. The broken wheel correctly identified its fault, except for one interval where its current inexplicably showed some variation. These errors could be overcome by tuning the transition probabilities, but that was not the point of this initial experiment.

Of the 50 test cases, a few also produced null distributions. This can arise from modeling errors; for example, there was no model for a crashed motor controller.

[4]This depends both on a constant bound on the set of transitions from any state and on the fact that each instance of a discrete state is in general distinct from other instances, so that state transitions form a chain without loops (except self-loops).

[5]Using straightforward multiplication for small matrices.

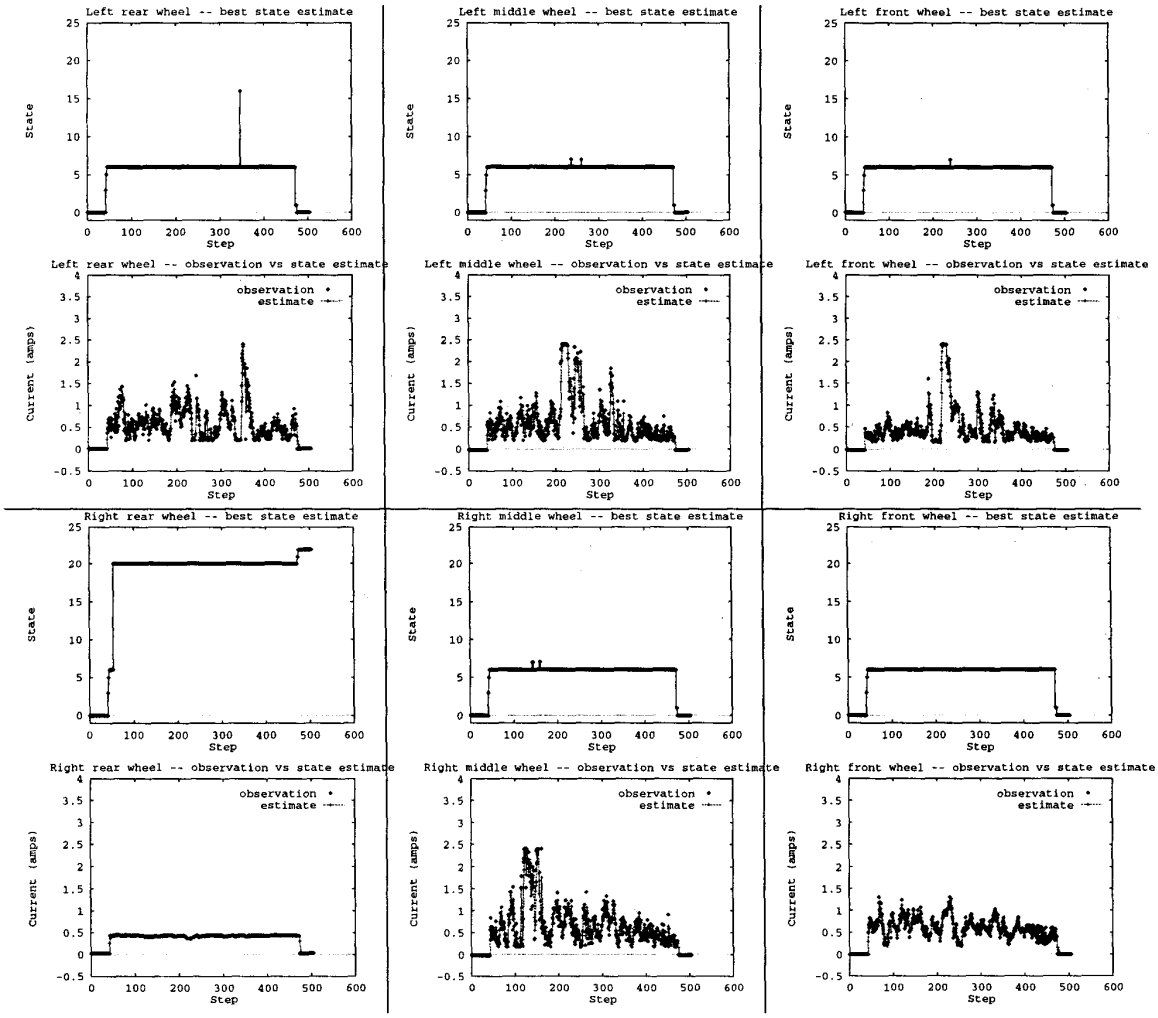

Figure 1: State estimates and wheel currents for each wheel, Experiment 1. The highest probability state is in fact the correct one except for a momentary misidentification in the left rear wheel.

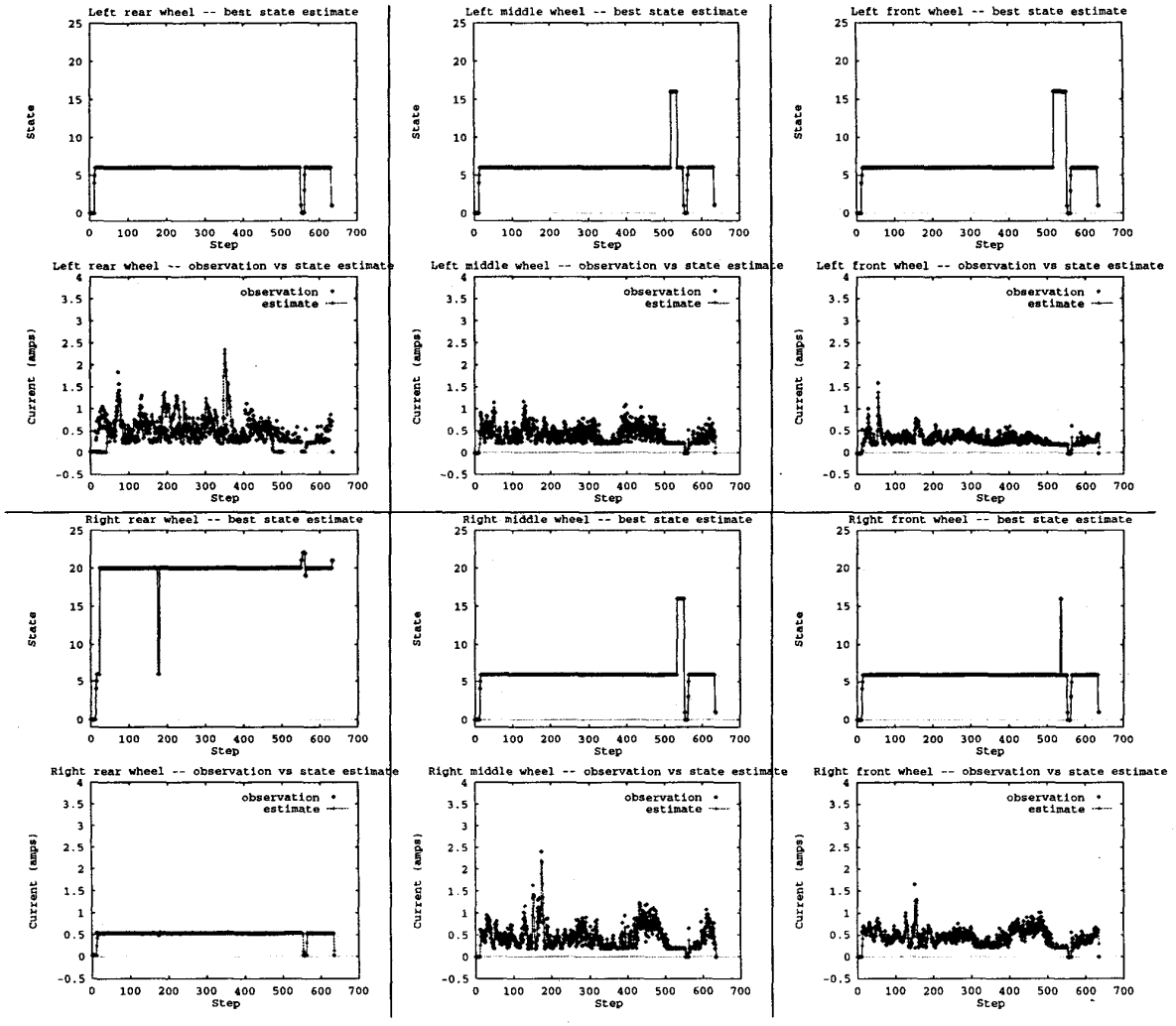

Figure 2: State estimates and wheel currents for each wheel, Experiment 2. Anomalies in the data give rise to state identification errors. A broken gear is found as the most likely state when the wheel currents flatten out. The broken wheel is momentarily labeled as working when the data vary unexpectedly.

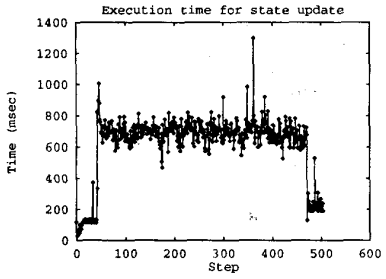

Figure 3: Elapsed time to update belief state for all 6 wheels, Experiment 1.

It can also be a result of the belief state truncation, which demonstrates the need to add mechanisms to avoid that (see Section 2.3).

The computational performance of the state identification program is promising. The prototype state identification system was coded in Java and tested on a Sun Ultra-2 (266 MHz), which is of the same order as our test rover platforms. The time required for the state estimation process in Experiment 1 can be seen in Figure 3. This is time to update all 6 wheels, for each of which there is a belief state of size 16 (hence 16 KFs per wheel, for a total of 96 KFs updated at each step). The update time is consistently just under a second while the rover is active, and less when idle.

4 Discussion

We have demonstrated an approach for computationally efficient state and fault identification that is particularly appropriate for the dynamic environment and noisy data encountered by an outdoor mobile robot. We envision this approach as a prototype for future planetary rovers, where accurate and efficient state identification is a critical element of long-term autonomous operation.

The approach presented here is a preliminary attempt to model and represent the states encountered by a rover. An obvious refinement is to develop more careful and complete models of rover operations. Although the crude models produce generally reliable state identification, they are imperfect; more precise models should lead to more accurate identification. Also, the KF model used for the prototype is a standard, discrete-time KF. This is a reasonable first approximation, but a more sophisticated KF model would support the more accurate models needed of the rover.

A KF model has a large number of parameters that can be adjusted. In the prototype, these were set to intuitively reasonable values by hand. These could be inferred from a larger corpus of experimental data. Additionally, the transition probabilities of the discrete states were equally hand-set. Anomalies such as the broken-wheel misidentification could be reduced by tweaking parameters, but ideally this would be information gathered from long-term experience with a platform (to gather reasonable fault probabilities).

As mentioned in Section 2.3, the approach needs to be extended to handle the case where a null distribution may arise. This should be a straightforward application of the work in [8].

Finally, the underlying probabilistic reasoning relies on a number of approximations. Understanding the relationship of the approximations to the complete, accurate probabilities is an important piece of understanding the quality of the approach as a whole.

References

[1] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proc. UAI*, 1998.

[2] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artif. Intelligence*, 32:100–117, 1987.

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

[4] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2), 1998.

[5] P. S. Maybeck. *Stochastic models, estimation, and control*, volume 1. Academic Press, 1979.

[6] N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams. Remote agent: To boldly go where no AI system has gone before. *Artif. Intelligence*, 103(1-2), 1998.

[7] H. E. Rauch. Intelligent fault diagnosis and control reconfiguration. *IEEE Ctl. Sys.*, 14(3), 1994.

[8] R. Washington. Making the impossible possible: Strategies for fast POMDP monitoring. In *Proc. of ECAI'98 Workshop on Monitoring and Control of Real-Time Intelligent Systems*, 1998.

[9] R. Washington. Markov tracking for agent coordination. In *Proc. Agents '98*, 1998.

[10] A. S. Willsky. A survey of design methods for failure detection in dynamic systems. *Automatica*, 12:601–611, 1976.