Demonstration of Prognostics-Enabled Decision Making Algorithms on a Hardware Mobile Robot Test Platform

Adam Sweet[1], George Gorospe[2], Matthew Daigle[1], José R. Celaya[2], Edward Balaban[1], Indranil Roychoudhury[2], and Sriram Narasimhan[3]

[1] *NASA Ames Research Center, Moffett Field, CA, 94035, USA*
{*adam.sweet, edward.balaban, matthew.j.daigle*}*@nasa.gov*

[2] *SGT Inc., NASA Ames Research Center, Moffett Field, CA, 94035, USA*
{*george.e.gorospe, jose.r.celaya, indranil.roychoudhury*}*@nasa.gov*

[3] *University of California, Santa Cruz, NASA Ames Research Center, Moffett Field, CA, 94035, USA*
sriram.narasimhan@nasa.gov

Abstract

Prognostics-enabled Decision Making (PDM) is an emerging research area that aims to integrate prognostic health information and knowledge about the future operating conditions into the process of selecting subsequent actions for the system. Previous work developing and testing PDM algorithms has been done in simulation; this paper describes the effort leading to a successful demonstration of PDM algorithms on a hardware mobile robot platform. The hardware platform, based on the K11 planetary rover prototype, was modified to allow injection of selected fault modes related to the rover's electrical power subsystem. The PDM algorithms were adapted to the hardware platform, including development of a software module framework, a new route planner, and modifications to increase the algorithms' robustness to sensor noise and system timing issues. A set of test scenarios was chosen to demonstrate the algorithms' capabilities. The modifications to run with a hardware platform, the test scenarios, and the test results are described in detail. The results show a successful use of PDM algorithms on a hardware test platform to optimize mission planning in the presence of electrical system faults.

1. Introduction

The research fields of system health, diagnostics, and prognostics have become mature to the point where the techniques have begun to be incorporated in new designs of aerospace vehicles (Reveley, Kurtoglu, Leone, Briggs, & Withrow, 2010). This has led to the newer research area called Prognostics-enabled Decision Making (PDM), which is devoted to the ability to incorporate system health information in making decisions in the planning and control of the system. A vehicle capable of making decisions, or assisting a human operator to make decisions, based on system health information could potentially accomplish more mission objectives, or operate with improved safety margins, than those that do not incorporate those considerations.

A useful way to drive maturation of algorithms in diagnostics and prognostics has been to develop test platforms where the algorithms may be evaluated. NASA Ames Research Center has developed several such test platforms, first in the electrical power system domain (Poll et al., 2007) and in the electromechanical actuator domain (Smith et al., 2009; Balaban et al., 2010). Each test platform has provided a means for controlled injection of faults to test the capabilities of the diagnostic and prognostic algorithms and has driven their development to be robust to real-world issues such as data latency and sensor noise. However, each test platform was designed primarily with the diagnostic and prognostic problems in mind. This led to the development of another test platform - the mobile robot test platform for testing and maturation of PDM algorithms.

Work began on a mobile robot test platform (Balaban et al., 2011, 2013) to provide a means for maturing PDM algorithms and verifying their predictions in a real-world environment. As described in previous publications, the mobile robot test platform is expected to support the following high-level tasks: (*i*) development of system-level and component-level PDM algorithms; (*ii*) development of realistic fault injection and accelerated aging techniques for algorithm testing; (*iii*) maturation and standardization of interfaces between reasoning algorithms; (*iv*) performance comparison of PDM algorithms from different sources; and (*v*) generation of publicly available datasets for enabling further PDM research. (Balaban et al., 2013) described the intended use of the test platform and the series of test scenarios which had been accomplished in simulation. This paper describes the adaptation of the algorithms to the hardware test platform, and the scenarios and results from using it to test PDM algorithms in the field.

The paper is organized as follows. Section 2 describes the platform modifications to support the new experiments. Section 3 presents the modifications to the PDM algorithms. Section 4 presents the experimental scenarios and results. Section 5 concludes the paper.

2. PLATFORM MODIFICATIONS

The ability to emulate realistic adverse events in the test platform is of key importance for the maturation process of PDM algorithms. In this context, an adverse event is regarded as an unexpected off-nominal physical change in the system under consideration. Such an event is to be properly observed by the health monitoring technology and properly mitigated or managed by the decisions and actions of the PDM system. Another important capability for a test platform is to provide a standard mechanism for its software modules to communicate with each other and with the PDM system. The adverse events emulated on the test platform and the software module framework are described in the sections below.

2.1. Hardware fault injection

The hardware faults currently implemented in the test platform are related to its electrical power system. As described in previous publications (Balaban et al., 2011, 2013), the rover vehicle under consideration is based on an electric power train in which the wheels are powered by electric motors and the power is stored in batteries. A variety of power conversion and mechanical faults in the electrical power train result in an increased power consumption in the form of higher levels of current demanded from the batteries. This ends up draining the batteries faster, thus potentially considerably reducing the duration of the rover mission. An example of an electrical power train fault that relates to increased energy consumption can be identified within the electrical motor controllers. A motor controller contains power switching elements like power transistors. The parasitic resistance of such devices and the lost of power dissipation capability due to degradation in performance during the device's lifetime, resulting in increased power consumption.

Because a variety of faults result in increased power consumption, the battery current drain circuit (parasitic load) was selected to implement on the robotic test platform. Other reasons for choosing that way of injecting hardware faults are that the circuit emulating the fault(s) has the ability to drain a

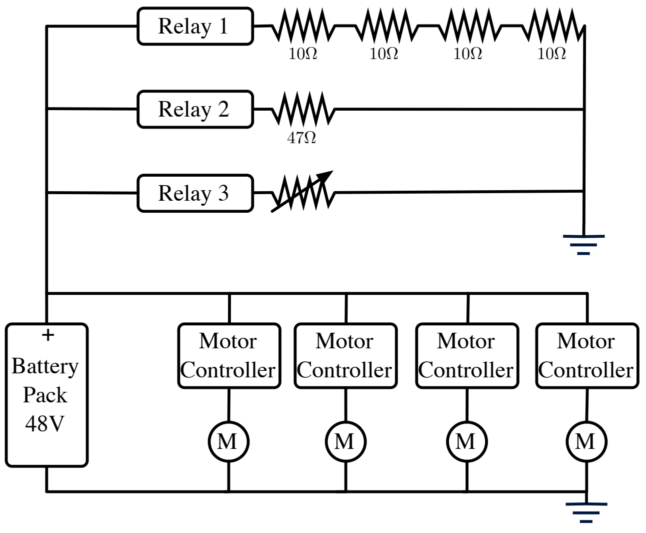


Figure 1. Battery current drain circuit schematic

variable amount of current and also that it is controlled programmatically. Figure 1 presents the battery current leakage circuit. It consists of three banks of resistors in parallel that can be engaged programmatically by closing the corresponding relay. The third bank in the diagram is a rheostat that is also controlled programmatically and ranges from 0 Ω to 10 Ω.

2.2. Sensor fault injection

Sensor fault injection is another method of introducing faults in the mobile robot platform. The prognostics and decision making components of the PDM system depend on accurate knowledge of the platform's state, in order to make accurate predictions and correct decisions based on those predictions. If a sensor is faulty and results in an incorrect estimate of the system's state, it could lead to either suboptimal decisions or, in the aviation domain, the potential loss of the mobile robot platform. Therefore, injecting sensor faults on the mobile robot platform is a useful way to test the PDM system's robustness and ability to ensure correct decisions are being made even in the presence of these types of faults.

Common types of sensor faults were described in (Balaban, Saxena, Bansal, Goebel, & Curran, 2009; Poll et al., 2011), and, in the course of this work, three types of sensor faults were implemented: stuck, offset, and drift. When a sensor is stuck, its value is set to a specified value and is unchanging thereafter. When a sensor has an offset fault, its value differs from the correct value by some specified constant amount. Finally, when a sensor has a drift fault, its value diverges slowly from the correct value over time. Examples of these are shown graphically in Figure 2.

2.3. Software module framework

The hardware test platform requires software to operate. The software consists of three major subcomponents: (*i*) the rover control and data acquisition module; (*ii*) the reasoning algorithms, (*iii*) and the communication infrastructure. The rover control and data acquisition software is implemented in LabVIEW (*LabVIEW version 12.0.0.4029*, 2012) and is responsible for interacting with the rover hardware. Control of the rover is performed by specifying wheel speeds for each individual rover wheel through the wheel motor controller hardware. Data acquisition is performed by multiple devices, and the LabVIEW control software is responsible for gathering the data from all the devices and making it available as a single sensor array. This data is sent to the PDM system.

The communication infrastructure is responsible for facilitating information sharing between the rover control software and the reasoning algorithms, as well as among the various reasoning algorithm modules. This is accomplished through a publish/subscribe architecture, which is implemented through the Internet Communication Engine, ICE (Henning, 2004). Standardized interface definition files are used to describe messages exchanged among the software and hardware modules. The message types include command inputs, sensor data, vehicle state information, fault diagnosis candidates, as well as unordered and ordered waypoint lists. A central server coordinates message exchanges among any number of devices on the same network. In order to be integrated into the architecture, a new reasoning module needs to only implement a minimal interface to register with the ICE server and to publish and/or subscribe to the appropriate messages. For example, a diagnostic module would subscribe to the rover commands and sensor data and, in turn, publish diagnostic messages. Thus the architecture allows for easy accommodation of modules implemented in different programming languages and running on dissimilar platforms.

3. ALGORITHM MODIFICATIONS

Modifications were also made to several PDM system algorithms; namely the state of charge estimator, the electrical power system (EPS) diagnoser, the route planner, and the decision maker. The changes made to each are described below.

3.1. State-of-Charge Estimator

The battery state-of-charge (SOC) estimator employs a model-based approach. Whereas in (Balaban et al., 2013) an electric circuit equivalent model of the battery cell was used, in this work the underlying model employed is an electrochemistry model of the lithium-ion cell presented in (Daigle & Kulkarni, 2013). The model has higher accuracy, yet is based only on ordinary differential equations, and, like the equivalent circuit model, can be simulated very quickly, suitable for real-time operations. As in (Balaban et al., 2013), the unscented Kalman filter (UKF) is used for state estimation (Julier & Uhlmann, 2004). The UKF estimates internal model states, from which SOC and the cell voltage are computed.

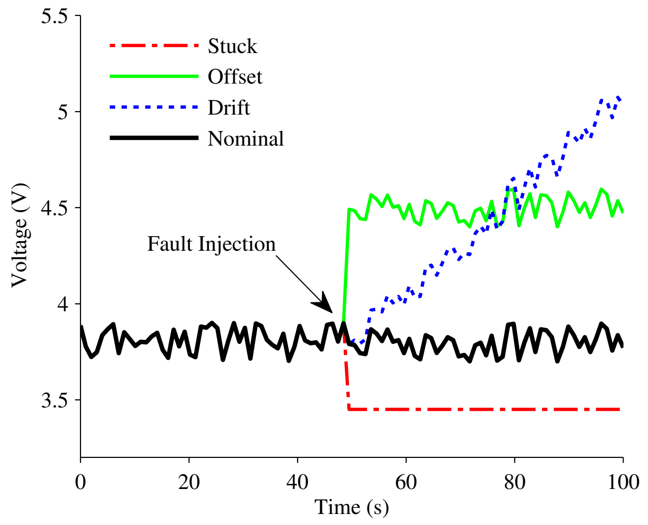


Figure 2. Examples of sensor faults

A distributed estimation approach (Daigle, Bregon, & Roychoudhury, 2014) can be used for the cells, where the local input to each cell's estimator is the measured battery current, i_B, divided by 2. Since the battery voltages on each parallel branch remain approximately balanced, the current going into the two branches is split evenly.

3.2. EPS Diagnoser

The diagnoser has three main diagnostic purposes, namely fault detection, isolation, and identification. Fault detection involves determining if a fault has occurred and is usually determined by taking the difference between the actual observed sensor readings and the model-predicted nominal behavior of these sensor readings, then determining if this difference is statistically significant. In order to compute the model-predicted signals, we adopt the structural model decomposition approach from (Roychoudhury, Daigle, Bregon, & Pulido, 2013) to decompose the global model of the EPS into smaller, local submodels, thus decomposing the model-based estimation problem. This is achieved by using measured sensor values as local inputs to the submodels. This, in fact, is what is done for the SOC estimators, formally justified by this decomposition approach. Thus, we obtain 25 local estimators, one for each cell voltage (using the SOC estimators), and one for the battery current (in which measured load current is used as an input, and we assume in the nominal case that the battery current is equal to the load current). The difference between a measured sensor value and the estimated value is termed the residual. A statistically significant deviation of a residual from zero indicates a fault. The employed fault detection method, based on a Z-test, is detailed in (Daigle et al., 2010) and is the same as used in (Balaban et al., 2013).

Once a fault is detected, a qualitative event-based diagnosis (QED) approach (Daigle, Roychoudhury, & Bregon, 2013) is invoked for fault isolation. For each available residual, a symbol generation routine is invoked that transforms a quantitative residual into qualitative symbols $\{0, +, -\}$ indicating whether or not the observed sensor reading is at, above, or below the estimated nominal value, respectively. Fault isolation is performed by comparing these observed symbols with the model-predicted symbols (Mosterman & Biswas, 1999; Daigle, Koutsoukos, & Biswas, 2009). Fault candidates that are inconsistent with the observed sequence of residual deviations are dropped. The fault candidate set is continually pruned as more residual deviations are observed until, ideally, the true single fault is the only fault candidate remaining[1]. Since the residuals are computed using local submodels, most faults affect only a few residuals (e.g., the parasitic load causes a deviation only in the battery current residual). This is in contrast to the global estimation approach in (Balaban et al., 2013) in which faults affect many residuals. Using the distributed estimation approach improves diagnosability (Daigle, Bregon, Biswas, Koutsoukos, & Pulido, 2012).

Once the true fault is isolated, a local model for estimating the fault parameters is used. The state estimate is augmented with the fault estimate for use in the prediction and decision-making steps. In the case of sensor faults, the faulty sensor value may have been used as a local input to an estimator, thus corrupting the resulting estimates. Therefore, once the sensor fault is identified, the estimators that used that (faulty) sensor value as an input must be reset to the time of fault occurrence and run again up to the current time using the corrected sensor value, so that a correct state estimate (to be used for prediction) can be obtained.

3.3. Route Planner

The route planner is a new component responsible for determining the route that the test vehicle takes. It operates on a set of waypoints, which represent points of scientific interest. Each waypoint consists of a location, specified with latitude, longitude, and altitude, and a reward, which is an integer representing the scientific importance of that waypoint. In the case of an aerial vehicle or a high-level simulation, direct paths between waypoints can often be assumed. This is not generally the case for a ground vehicle, where terrain features and obstacles need to be taken into account when planning vehicle movement. The available waypoints are defined in advance and are located at the street intersections in the experiment's geographical area, as shown in Figure 3a. Not all of the defined waypoints were used as primary waypoints in the experiments described later; the choice of waypoints is described in Section 4. The waypoints which were used are shown as green in Figure 3a and the unused waypoints are shown in black. The waypoints are identified with numbers, shown in the figure just after the letter 'W' (for "waypoint"). In Figure 3a, the reward value of each waypoint is shown in parentheses after the waypoint number. Given the set of waypoints, the route planner calculates routes for all possible pairs of waypoints going in either direction. A route between any two waypoints is approximated as a set of linear segments between secondary waypoints. This set is translated into a list of tuples $\{\textit{heading}, \textit{distance}, \textit{elevation change}\}$, with each tuple providing instructions on getting from one secondary waypoint to the next.

The route planner uses the Google Maps API (*JavaScript Google Maps Application Programming Interface, version 3.0*, 2014) to calculate the routes. The route planner then considers all pairs of waypoints (in both directions). For each pair of primary waypoints, the Google Maps API is used to identify the secondary waypoints between the primary waypoints. The API provides latitude, longitude, and altitude for the secondary waypoints as a sequence of steps to get from the source waypoint to the destination waypoint. The planner then steps through the secondary waypoints in order and uses the API to determine the heading between the last waypoint and current waypoint. It also calculates the altitude change based on the already retrieved altitudes for the waypoints. The result is a three-dimensional array where the first and second dimension indicate pairs of primary waypoints. The third dimension is used to list routes to secondary waypoints in order, resulting in the aforementioned list of tuples.

3.4. Decision Maker

The decision-making algorithm used in this work, shown in Algorithm 1, is similar to the one presented in (Balaban & Alonso, 2013). It is based on a particle-filtering pattern (Gordon, Salmond, & Smith, 1993) and is summarized below.

Algorithm 1 uses a set of k particles, where each particle p_i is initialized with the starting waypoint wp_1 and assigned a uniform weight of $w_i = 1/k$. The starting waypoint is the waypoint where the vehicle is located at the point of executing the algorithm (not necessarily the original starting point of the route). For simplicity of explanation, the algorithm presented here operates over one-way paths, where the starting waypoint is not always the same as the ending waypoint. In the actual implementation, a choice between one-way and round-trip routes is implemented via a straightforward extension.

During each of the iterations of the algorithm (and for each particle), the path associated with a particle is sampled ran-

[1]This work is restricted to single faults only.

Algorithm 1 PF

```
1: inputs: $\{wp_i\}_{i=1}^{N}, K$
2: outputs: $p^*$
3: $p_1, \ldots, p_K \leftarrow \{wp_1\}$
4: $w_1, \ldots, w_K \leftarrow 1/k$
5: for $d \leftarrow 1, D$ do
6:     for $k \leftarrow 1, K$ do
7:         $\tau \leftarrow permute(\{wp_i\}_{i=1}^{N} - p_k)$
8:         $l \leftarrow -1$
9:         repeat
10:            $l \leftarrow l + 1$
11:            $p_{test} = \{p_k, \{wp_1, \ldots, wp_l\}\}$
12:            $\{b, \mathcal{R}, \mathcal{C}_h\} \leftarrow simulate(p_{test})$
13:            $w_k \leftarrow \{\Theta_R, \Theta_h\} \cdot \{R, -C_h\}^T$
14:        until $\mathcal{F}(b) = true$
15:        if $l \geq 1$ then
16:            $p_k \leftarrow \{p_k, \{wp_1\}_\tau\}$
17:        end if
18:    end for
19:    $j \leftarrow \arg\max_j w_j$
20:    $p^* \leftarrow p_j$
21:    $\{w_1, ..., w_K\} \leftarrow \{w_1, ..., w_K\} / \sum_{i=1}^{K} w_i$
22:    $\{p_1, ..., p_K\} \leftarrow resample(\{p_1, ..., p_K\}, \{w_1, ..., w_K\})$
23: end for
```

domly out of the set of unvisited waypoints up to the maximum length of N. Each sample is tested in the simulator and the particle weight updated proportionally to the objective function value (which incorporates path costs in addition to rewards). Unless system failure is believed to be likely for even the shortest path extensions, the particle path is extended by one waypoint (the first one in the randomized remaining waypoints set τ).

The number of algorithm iterations, D, is equal to N for the deterministic simulator mode and can be set to $D > N$ otherwise, to help prevent potentially promising particles from being ruled out too early. The highest weight particle is identified and stored after each iteration, to enable interruptibility. Particle weights are then normalized and the particles are resampled. The overall computational complexity of the algorithm is $O(N^2)$.

The objective function used to guide search of the solution space is the following:

$$J = \{\Theta_R, \Theta_h\} \cdot \{R, -C_h\}^T, \tag{1}$$

where $\mathcal{R}$ is the expected cumulative reward along a route, C_h is the correspondent expected health cost, Θ_R and Θ_h are the weights for rewards and health costs, respectively. The simulator used with the PF algorithm utilizes a simplified power consumption model of the rover. A candidate route is divided into linear and turning segments and the resulting list of segments is processed sequentially. For the straight route segments, the following relationship was used to estimate the current drawn from the batteries:

$$i_l = \frac{mgv}{\eta_e E}(\sin\alpha + \mu cos\alpha), \tag{2}$$

where i_l is the linear segment current, η_e is the electrical transmission efficiency coefficient, E is the bus voltage, m is the mass of the rover, g is the acceleration of gravity, v is the magnitude of the linear velocity, α is the incline angle, and μ is the coefficient of surface friction. For this set of experiments linear velocity was kept constant. For the turning segments, a constant rate of turn ω was assumed, associated with a constant current draw i_t. When evaluating a candidate route, a discrete time simulation is performed (with the time step dt normally set to $1s$), taking into account the nonlinear relationship between current draw at a particular instance in time and the corresponding drop in battery cell voltage (and in the SOC of the battery cells).

The battery model used in the simulator is described in (Daigle, Saxena, & Goebel, 2012). The parameters used with the model remained the same as in the aforementioned paper. This equivalent circuit model was integrated and tested with the decision maker prior to the newer electrochemistry model (Daigle & Kulkarni, 2013) becoming available and will be updated to the latter in the near future. The rest of the model parameters used in the experiments are the following are described in Table 1.

Table 1. DM model parameters used in the experiments

Parameter	Value	Units
m	150.0	kg
v	0.4	m/s
ω	0.07	rad/s
μ	0.06	
i_t	5.0	A
η_e	0.8	

A set of $K = 50$ particles was used by PF algorithm. The values of objective function weights used were $\Theta_R = 0.9$ and $\Theta_h = 0.1$.

4. EXPERIMENTS AND RESULTS

The PDM system described above was demonstrated in the field though a set of scenario-based experiments. The details of the scenarios, including the number of waypoints, the overall distance, the distances between waypoints, the fault injection location and fault magnitude were chosen to clearly show the capabilities of the PDM algorithms. The overall distance of the nominal trajectory would have to be such that the vehicle would be capable of completing it fully without system faults. That full nominal scenario trajectory was divided and arranged into waypoints, chosen to ensure the potential for the system to replan its trajectory before a potential system fault would cause the vehicle to reach the end of useful life. Several fault scenarios were chosen as well, where the PDM system was expected to optimize the trajectory while ensuring the safe return of the vehicle. The location at which the fault is injected and its magnitude were chosen to allow the cumulative effects of the fault to cause the end of the system's remaining useful life before it reached the final waypoint. These experimental scenarios and the results of each are described below.

4.1. Nominal

The nominal scenario consists of 5 waypoints, and no fault was injected during this scenario. The vehicle began at waypoint 9 and traveled to waypoints 2, 5, 7, and back to 9. These waypoints are shown in Figure 3b; the unused waypoints are not shown for clarity. The reward value used for waypoint 9 is 70, and for waypoints 2, 5, and 7 are 30, 90, and 20, respectively. The route planner inserted secondary waypoints as required to navigate this route, where secondary waypoints have no reward.

In this scenario, the vehicle successfully followed the nominal route, covering a distance of approximately 970 m, and gained the reward for all of the waypoints, for a total reward of 280. The nominal route is shown as the blue line in Figure 3b, generated from the vehicle's global positioning system (GPS) sensor values recorded during the scenario. At the end of the nominal scenario the batteries had an estimated SOC of 57.6%. Note that even in the nominal case, the PDM system ran and determined that it was feasible to achieve all of the given waypoints.

4.2. Battery Parasitic Load Fault without PDM

As a second scenario, a battery parasitic load fault (as described in Section 2.1) was injected during the route traversal, with PDM system was not running. This scenario also began and ended at waypoint 9 and consisted of the same waypoints as for the nominal scenario, with the same reward values. However, shortly before reaching waypoint 2, a battery parasitic load fault was injected into the electrical system of the vehicle. This is shown in Figure 3c. The first two relays were activated, resulting in an equivalent parasitic resistance of $21.6\ \Omega$ (see circuit diagram in Figure 1) and an increased current draw from the batteries.

In this scenario, with the battery parasitic load active and following the nominal route, the vehicle ran out of power before returning to the starting waypoint. The route is shown in red in Figure 3c, and the location where the vehicle ran out of power is marked. This scenario showed that the nominal route is, in fact, infeasible under the battery drain fault.

4.3. Battery Parasitic Load Fault with PDM

As a third scenario, a battery parasitic load fault was again injected (resulting in the same parasitic resistance of $21.6\ \Omega$) shortly before reaching waypoint 2, while following the same waypoints as the nominal scenario. However, in this case the PDM system was enabled.

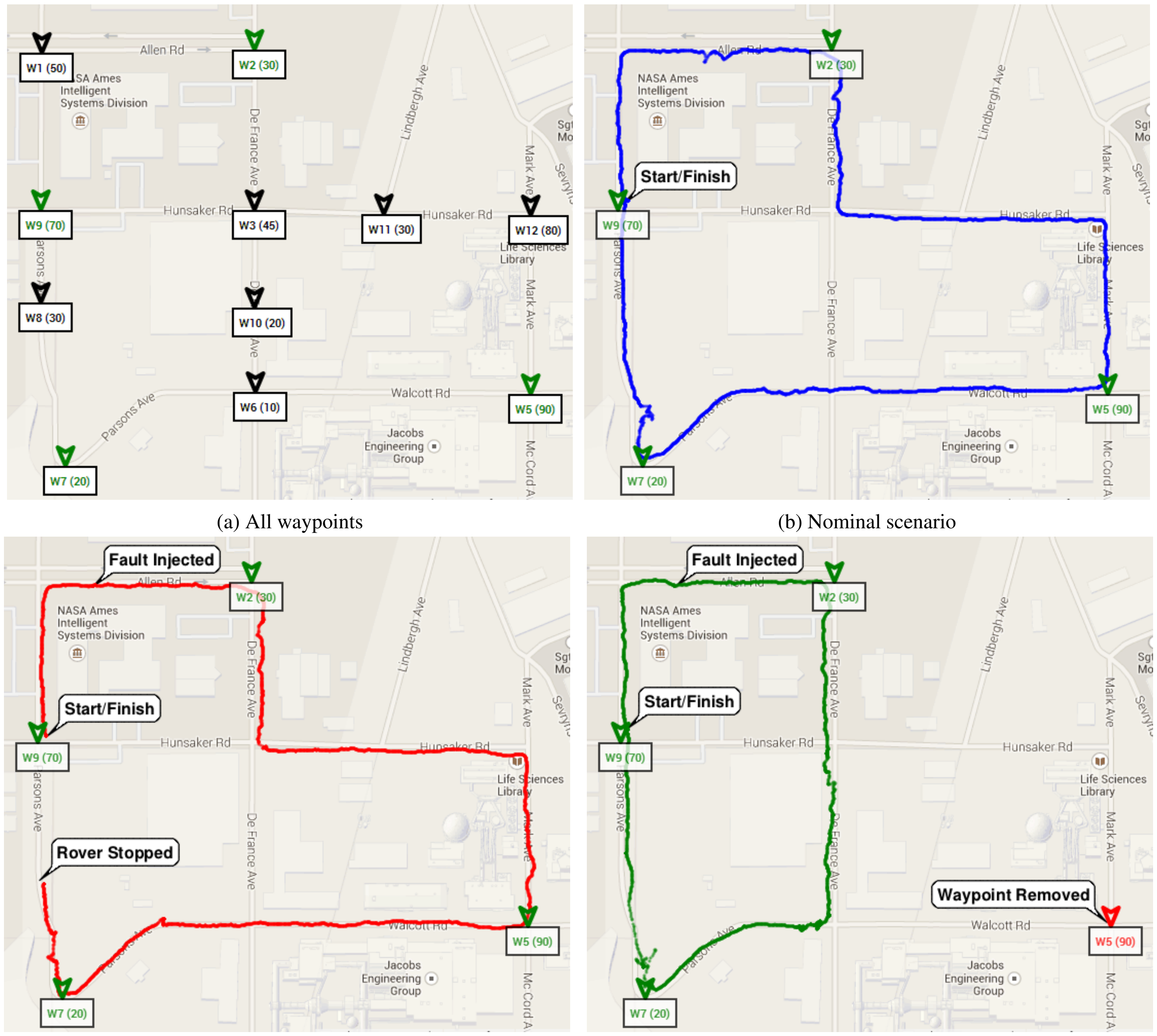
In this scenario, the EPS diagnoser detected that the battery parasitic load has been injected and estimated the equivalent resistance value. It reported its estimate of the equivalent resistance value 14 s after the fault was injected. The estimated resistance was $19.5\ \Omega$, which is an error of only 9.7% from the actual parasitic resistance of $21.6\ \Omega$. The EPS diagnoser then sent that estimated parasitic resistance value to the decision maker along with the battery SOC estimate. When the vehicle arrived at waypoint 2, the decision maker used the information from the EPS diagnoser and determined that the vehicle's original route is no longer feasible. It then performed an optimization to determine a new route which maximized the overall reward for the scenario, while ensuring that the vehicle can return safely to the starting point. As can be seen in Figure 3d, the PDM system eliminated waypoint 5 (shown in red), but kept waypoint 7. The alternative route taken, shown on the figure in green, covered a distance of approximately 713 m. The vehicle successfully navigated the new route and returned to the starting waypoint 9, for a total reward of 190. At the end of the scenario the estimated SOC of the batteries was 14.5%.

Note that a conservative option existed: to return to the starting waypoint as soon as possible after the fault was detected. However, that route would only have gained a total reward of 170. It would not have made optimal use of the vehicle's remaining useful life and, therefore, was not chosen by the PDM system.

4.4. Bus Current Sensor Fault with PDM

As a fourth scenario, a bus current sensor fault was injected (also just before reaching waypoint 2) while following the same waypoints as the nominal scenario. The bus current sensor value was overridden to always report a value of 0.0 A.

In this scenario, the EPS diagnoser detected that the current sensor is faulty and reported that to the decision maker. The decision maker performed an optimization with this vehicle state and the given waypoints and determined that the vehicle is able to complete the original route given the fault. Therefore, it did not modify the vehicle route. Since the mission was unmodified, the vehicle traversed the same route as in the nominal scenario shown in Figure 3b, for the same total reward of 280. At the end of the scenario the estimated SOC of the batteries was 70.8%.



(a) All waypoints

(b) Nominal scenario

(c) Battery parasitic load fault scenario without PDM

(d) Battery parasitic load fault scenario with PDM

Figure 3. Vehicle route taken in nominal and battery parasitic load scenarios with and without PDM

5. CONCLUSIONS

This paper described a successful demonstration of PDM algorithms running onboard a hardware mobile robot test platform. The demonstration required modifications to both the platform and the algorithms. The demonstrations took the form of a set of challenge scenarios. The data files from these scenarios will be made available for download, to allow testing of other prognostic and PDM algorithms.

Planned future work involves deployment of PDM algorithms on an unmanned aerial vehicle, incorporation of uncertainty estimates in the reported health parameters, and implementation of additional faults in the hardware test platform. Possible future work also includes modifications to support different types of decision-making, such as adapting parameters and constraint relaxation in the PDM optimization. As the PDM algorithms are further developed, this robotic test platform will be modified to continue to evaluate them in a real-world setting.

ACKNOWLEDGMENTS

The authors would like to gratefully acknowledge the contributions of researchers and student interns at NASA Ames Research Center: Kai Goebel, Brian Bole, Shankar Sankararaman, and Sebastian Hening. The funding for this research is provided by NASA ARMD System-wide Safety & Assurance Technology (SSAT) project.

REFERENCES

Balaban, E., & Alonso, J. J. (2013). A Modeling Framework for Prognostic Decision Making and its Application to UAV Mission Planning. In *Annual conference of the prognostics and health management society* (pp. 1–12). New Orleans, LA.

Balaban, E., Narasimhan, S., Daigle, M., Celaya, J., Roychoudhury, I., Saha, B., ... Goebel, K. (2011, September). A mobile robot testbed for prognostics-enabled autonomous decision making. In *Annual conference of the prognostics and health management society 2011.* Montreal, Canada.

Balaban, E., Narasimhan, S., Daigle, M., Roychoudhury, I., Sweet, A., Bond, C., & Gorospe, G. (2013). Development of a mobile robot test platform and methods for validation of prognostics-enabled decision making algorithms. *International Journal of Prognostics and Health Management*, *4*(1).

Balaban, E., Saxena, A., Bansal, P., Goebel, K. F., & Curran, S. (2009). Modeling, detection, and disambiguation of sensor faults for aerospace applications. *Sensors Journal, IEEE*, *9*(12), 1907–1917.

Balaban, E., Saxena, A., Narasimhan, S., Roychoudhury, I., Goebel, K., & Koopmans, M. (2010, September). Airborne electro-mechanical actuator test stand for development of prognostic health management systems. In *Annual conference of the prognostics and health management society 2010.* Portland, OR.

Daigle, M., Bregon, A., Biswas, G., Koutsoukos, X., & Pulido, B. (2012, August). Improving multiple fault diagnosability using possible conflicts. In *Proceedings of the 8th ifac symposium on fault detection, supervision and safety of technical processes* (p. 144-149).

Daigle, M., Bregon, A., & Roychoudhury, I. (2014). Distributed prognostics based on structural model decomposition. *IEEE Transactions on Reliability*.

Daigle, M., Koutsoukos, X., & Biswas, G. (2009, July). A qualitative event-based approach to continuous systems diagnosis. *IEEE Transactions on Control Systems Technology*, *17*(4), 780–793.

Daigle, M., & Kulkarni, C. (2013, October). Electrochemistry-based battery modeling for prognostics. In *Annual conference of the prognostics and health management society 2013* (p. 249-261).

Daigle, M., Roychoudhury, I., Biswas, G., Koutsoukos, X., Patterson-Hine, A., & Poll, S. (2010, September). A comprehensive diagnosis methodology for complex hybrid systems: A case study on spacecraft power distribution systems. *IEEE Transactions of Systems, Man, and Cybernetics, Part A*, *4*(5), 917–931.

Daigle, M., Roychoudhury, I., & Bregon, A. (2013, October). Qualitative event-based diagnosis with possible conflicts: Case study on the fourth international diagnostic competition. In *Proceeedings of the 24th international workshop on principles of diagnosis* (p. 230-235).

Daigle, M., Saxena, A., & Goebel, K. (2012). An Efficient Deterministic Approach to Model-based Prediction Uncertainty Estimation. In *Annual conference of the prognostics and health management society.*

Gordon, N. J., Salmond, D. J., & Smith, A. F. (1993). Novel Approach to Nonlinear/non-Gaussian Bayesian State Estimation. *IEE Proceedings F (Radar and Signal Processing)*, *140*(2), 107–113.

Henning, M. (2004). A new approach to object-oriented middleware. *IEEE Internet Computing*, *8*(1), 66–75.

JavaScript Google Maps Application Programming Interface, version 3.0. (2014). Mountain View, California: Google Corporation.

Julier, S. J., & Uhlmann, J. K. (2004, March). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, *92*(3), 401–422.

LabVIEW version 12.0.0.4029. (2012). Austin, Texas: National Instruments Corporation.

Mosterman, P. J., & Biswas, G. (1999). Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, *29*(6), 554-565.

Poll, S., de Kleer, J., Abreau, R., Daigle, M., Feldman, A., Garcia, D., ... Sweet, A. (2011, October). Third international diagnostics competition – DXC'11. In *Proc. of the 22nd international workshop on principles of diagnosis* (pp. 267–278).

Poll, S., Patterson-Hine, A., Camisa, J., Nishikawa, D., Spirkovska, L., Garcia, D., ... others (2007, May). Evaluation, selection, and application of model-based diagnosis tools and approaches. In *Aiaa infotech@aerospace 2007 conference and exhibit.*

Reveley, M. S., Kurtoglu, T., Leone, K. M., Briggs, J. L., & Withrow, C. A. (2010, December). *Assessment of the state of the art of integrated vehicle health management technologies as applicable to damage conditions* (TM No. 2010-216911). Cleveland, OH: NASA.

Roychoudhury, I., Daigle, M., Bregon, A., & Pulido, B. (2013, March). A structural model decomposition framework for systems health management. In *Proceedings of the 2013 ieee aerospace conference.*

Smith, M., Byington, C., Watson, M., Bharadwaj, S., Swerdon, G., Goebel, K., & Balaban, E. (2009). Experimental and analytical development of health management for electro-mechanical actuators. In *Ieee aerospace conference* (pp. 1–14).

BIOGRAPHIES

Adam Sweet is a research engineer in the Diagnostics and Prognostics group at NASA Ames Research Center. He graduated with an MS in Mechanical Engineering from UC Berkeley in 1999, and has worked at Ames ever since. His project experience encompasses robotics, hybrid system simulation, model-based diagnosis, and flight software development for nanosatellites.

George Gorospe rreceived the B.E. degree in Mechanical Engineering from the University of New Mexico, Albuquerque, New Mexico, USA, in 2012. Since October 2012, he has been a research engineer at NASA Ames Research Center. In May 2013 he joined Stinger Ghaffarian Technologies and the Prognostic Center of Excellence at NASA Ames Research Center. His current research interests include space mission design, systems engineering, and autonomous mobile robot control and control systems design.

Matthew J. Daigle received the B.S. degree in Computer Science and Computer and Systems Engineering from Rensselaer Polytechnic Institute, Troy, NY, in 2004, and the M.S. and Ph.D. degrees in Computer Science from Vanderbilt University, Nashville, TN, in 2006 and 2008, respectively. From September 2004 to May 2008, he was a Graduate Research Assistant with the Institute for Software Integrated Systems and Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN. From June 2008 to December 2011, he was an Associate Scientist with the University of California, Santa Cruz, at NASA Ames Research Center. Since January 2012, he has been with NASA Ames Research Center as a Research Computer Scientist. His current research interests include physics-based modeling, model-based diagnosis and prognosis, simulation, and hybrid systems.

José R. Celaya is a research scientist with SGT Inc. at the Prognostics Center of Excellence, NASA Ames Research Center. He received a Ph.D. degree in Decision Sciences and Engineering Systems in 2008, a M. E. degree in Operations Research and Statistics in 2008, a M. S. degree in Electrical Engineering in 2003, all from Rensselaer Polytechnic Institute, Troy New York; and a B. S. in Cybernetics Engineering in 2001 from CETYS University, México.

Edward Balaban is a research engineer in the Diagnostics and Prognostics group at NASA Ames Research Center. He received the Bachelor degree in Computer Science from The George Washington University in 1996 and the Master degree in Electrical Engineering from Cornell University in 1997. His main areas of interest are diagnostics, prognostics, and prognostics-enabled decision making. During his years at Ames he has participated in the research and development of system health management elements for the X-34 experimental reusable launch vehicle, the International Space Station, robotic astronaut assistants, autonomous planetary drills, and other aerospace applications. He is a member of the PHM Society, the IEEE, and the AIAA.

Indranil Roychoudhury received the B.E. (Hons.) degree in Electrical and Electronics Engineering from Birla Institute of Technology and Science, Pilani, Rajasthan, India in 2004, and the M.S. and Ph.D. degrees in Computer Science from Vanderbilt University, Nashville, Tennessee, USA, in 2006 and 2009, respectively. Since August 2009, he has been with SGT, Inc., at NASA Ames Research Center. Dr. Roychoudhury is a member of the Prognostics and Health Management Society and the IEEE. His research interests include hybrid systems modeling, model-based diagnostics and prognostics, distributed diagnostics and prognostics, and Bayesian diagnostics of complex physical systems.

Sriram Narasimhan is a Project Scientist with University of California, Santa Cruz working as a contractor at NASA Ames Research Center in the Discovery and Systems Health area. He received his M.S and Ph.D. in Electrical Engineering and Computer Science from Vanderbilt University. He also has a M.S in Economics from Birla Institute of Technology and Science. His research interests are in model-based diagnosis with a focus on hybrid and stochastic systems. He is the technical lead for the Hybrid Diagnosis Engine (HyDE)