Embedding Health Management into Mission Tasking for UAV Teams

Mario Valenti, Brett Bethke, Jonathan P. How, Daniela Pucci de Farias, and John Vian

Abstract— **Coordinated multi-vehicle autonomous systems can provide incredible functionality, but off-nominal conditions and degraded system components can render this capability ineffective. This paper presents techniques to improve mission-level functional reliability through better system self-awareness and adaptive mission planning. In particular, we extend the traditional definition of health management, which has historically referred to the process of actively monitoring and managing vehicle sub-systems (e.g., avionics) in the event of component failures, to the context of multiple vehicle operations and autonomous multi-agent teams. In this case, health management information about each mission system component is used to improve the mission system's self-awareness and adapt vehicle, guidance, task and mission plans. This paper presents the theoretical foundations of our approach and recent experimental results on a new UAV testbed.**

I. Introduction

Recently, unmanned vehicle mission systems have become extremely important in aiding military and civilian organizations for gathering information and assessing situations from remote locations. As such, teams of unmanned vehicles are being used to gather data from different locations, thus providing operators with the most up-to-date situational information for the areas they cover. These multi-vehicle autonomous teams can provide incredible functionality, but off-nominal conditions and degraded system components can render their capabilities ineffective.

During a mission, how should teams of autonomous agents be managed to meet scenario objectives while minimizing the total cost of an operation?

This question directly relates to the health of the mission system. In the past, the term "health management" was used to define systems that actively monitored and managed vehicle sub-systems (e.g., flight controls, fuel management, avionics) in the event of component failures [1]. This definition can be extended to the context of multiple vehicle operations and autonomous multi-agent teams where multi-team mission groups serve as a "vehicle system," multi-agent teams are sub-systems to the larger mission team, and each vehicle is a sub-system of each multi-agent team. As with mission-critical systems for a single agent, multi-agent task allocation and mission management systems must account for vehicle- and system-level health-related issues to ensure that these systems are cost effective to operate over the duration of a mission. While this problem may be formulated as a very large mathematical programming problem, this approach is likely to be computationally intractable for any computer or human operator to solve in real-time (even for a problem of reasonable size).

Fig.1: Eight autonomous UAVs shown in the MIT RAVEN

Although many researchers have been discussing autonomous multi-agent operations [2], [3], [4], [5], more work is needed on how to perform multi-agent health management for autonomous task groups. In principle, some of the issues raised in this problem are similar to questions arising in manufacturing systems [6], [7] and air transportation [8], [9]. There are three major differences between previous and ongoing research and our work. First, this research considers issues related to vehicle health management (e.g., refueling, vehicle failures). Our goal is to ensure that each agent safely returns to the base location after a task is completed, thus reducing operating costs over multiple missions. Second, this work considers issues vital to continuous (24–7) operations which include operator and shift changes, vehicle maintenance periods, and extended theater operations. Third, we present flight test results using these planning techniques in multi-vehicle experiments using the RAVEN (Real-time Autonomous Vehicle indoor test ENvironment) in the MIT Aerospace Controls Laboratory (Figure 1). This testbed was designed to enable continuous (24–7) operation of an autonomous swarm of more than 10 UAVs.

II. Health Management Techniques for the Multi-Vehicle Mission Problem

The multi-vehicle mission problem is very complex, so we first consider health management techniques that can be added to existing mission problem formulations to improve system performance during mission operations. First, note that most planning techniques are decomposed into a

Ph.D. Candidate, EECS Dept., Massachusetts Institute of Technology, Cambridge, MA 02139 valenti@mit.edu

S.M. Candidate, Dept. of Aeronautics and Astronautics, MIT bbethke@mit.edu

Professor of Aeronautics and Astronautics, MIT, jhow@mit.edu

Assistant Professor of Mechanical Engineering, MIT, pucci@mit.edu

Technical Fellow, Boeing Phantom Works, Seattle, WA john.vian@boeing.com

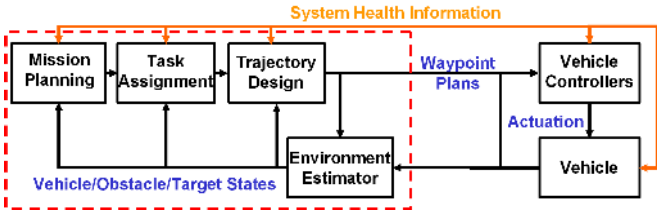

Fig. 2: Health management-based hierarchical architecture model for multi-vehicle mission systems

multi-tiered architecture where missions, tasks and vehicle-specific activities are managed by different components to meet real-time computational deadlines. Similar architectures have been used in other multi-vehicle systems and testbeds (see references in [10], [11]); however, most systems lack health monitoring components to evaluate the subsystem performance. As a result, adding system health monitors to a simple hierarchical design (as shown in Figure 2) can improve a system's run-time capabilities by ensuring that it maintains a basic, functional capability during a mission in accordance with system goals.

In using the hierarchical architecture as shown in Figure 2, a number of implementation concerns arise that can reduce performance and possibly jeopardize the success of a mission in a real-time environment. Some of these real-time challenges include communication, computational, and control issues, human/operator interaction, integration issues, safety and system health concerns among many others. In addition, external and environmental conditions also affect system performance. In most real-life situations these conditions are not controlled and the operational environment is only partially known. Therefore, as explained in [12], real-time implementations of mission system technology must be robust to external and unexpected conditions. Systems lacking vehicle and system component performance feedback loops become reactionary to problems that occur in a mission system, which can cause a reduction in system performance. For this reason, health management feedback and monitoring in a mission system is essential in maintaining a proactive approach to mission planning.

III. Vehicle Level Health Management Modules

First, subsystems (e.g., the vehicles) in the mission system must contain health monitors to provide status information to aid in higher level decision making. For example, most vehicles have flight time limitations based on fuel and maintenance constraints. This concern is a very important for small, electric-powered air helicopters and quadrotors where vehicle endurance scales with battery size, and thus battery weight (which has a negative effect of vehicle endurance). Although battery technology has improved in recent years (for example, a 3-cell, 11.1 V 1320 mAh Li-Poly battery from Thunder Power Batteries [13] is approximately 100 gms), an electric powered air vehicle's flight time is largely impacted by the vehicle's lift capabilities. In addition, the average flight time of electric-powered helicopters and quadrotors (such as the Draganflyer V Ti Pro [14]) are limited by the motors used and desired payload capacity.

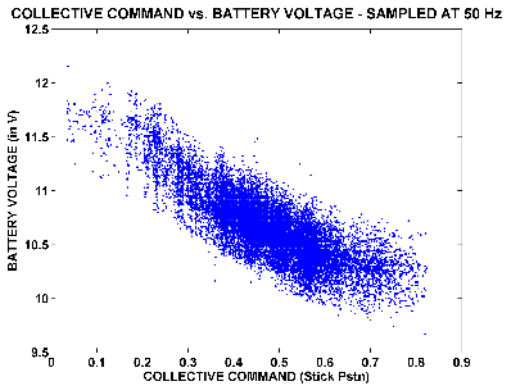

Fig. 3: Comparison between battery voltage and collective stick position during a hover test

As part of our testing, the relationships between power usage, battery charge and other parameters where examined for a set of off-the-shelf electric-powered quadrotor vehicles in the MIT RAVEN [10]. This testing showed that there is a strong correlation between the battery voltage and the average collective stick position value. As the voltage of the battery decrease over time (due to battery use), the collective stick command is increased (as shown in Figure 3). Figure 4 shows that as the battery's charge level decreases, the average collective command must increase over time for the vehicle to maintain its current position. Notice that for this vehicle (using a 11.1 V 1320 mAh battery, flying without a camera), the collective command increases rapidly initially during take-off of the vehicle and steadily increases almost linearly until the vehicle's battery begins to rapidly lose charge near the end of the flight. Therefore, using experimental data, a model (based on real flight data) could be generated using the vehicle's current altitude and collective position to estimate the vehicle's remaining flight time *non-invasively*. Here, the model does *not* use sensors on the vehicle to generate this estimate of remaining flight time. This model was tested against actual vehicle hover flights to determine whether the model could be used in practice.

For this task, a support vector regression (SVR) model (based on real flight data) was generated using the vehicle's altitude and collective position to estimate the vehicle's remaining flight time. The main idea behind an SVR model is that a model is created (using experimental data) from input-output pairs that can be used to predict the output to a system given an input without explicitly knowing the model of the system [15], [16], [17]. A model of the form

$$f(x) = \sum_{i \in SV} (\bar{\alpha}_i - \bar{\alpha}_i^*) K(x, x_i) + \bar{b} \qquad (1)$$

is used to generate a non-linear mapping into a higher-dimensional feature space of the form used to perform a regression on any data point $x \in \mathbb{R}^n$ [16]. Here, $K(x, x_i)$ is defined as the kernel function which generates the inner product in the high-dimensional feature space and $\bar{b}$ is a constant [15]. In addition, the weighting parameters $\bar{\alpha}_i^*$, $\bar{\alpha}_i$ for all $i \in SV$ for the ε-insensitive SVR problem are

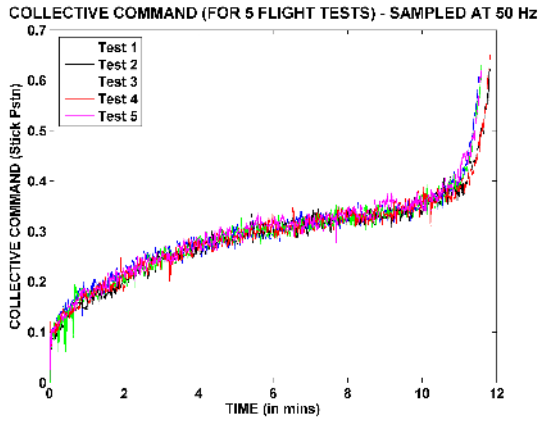

Fig.4: Collective stick pstn vs time during five hover tests

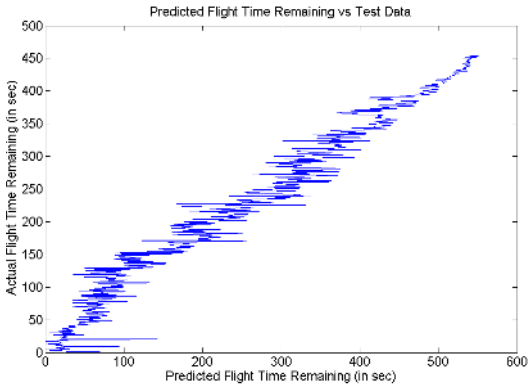

Fig.5: Predicted vs actual remaining flight for a hover test

determined by solving the optimization problem [16]:

$$\begin{aligned} \max_{\alpha,\ \alpha^*} \quad & \sum_{i=1}^{r} \alpha_i^*(y_i-\varepsilon)-\alpha_i(y_i+\varepsilon) \\ & -\tfrac{1}{2}\textstyle\sum_{i,j}^{r}(\alpha_i^*-\alpha_i)(\alpha_j^*-\alpha_j)K(x_i,x_j) \quad (2) \\ \text{subj. to} \quad & \sum_{i=1}^{r}(\alpha_i-\alpha_i^*)=0, \quad \forall i\in\{1,\ldots,r\} \\ & 0\geq\alpha_i,\ \alpha_i^*\geq C \quad \forall i\in\{1,\ldots,r\} \end{aligned}$$

where $\{(x_1,y_1),(x_2,y_2),...,(x_r,y_r)\}$ is the set of training pairs introduced into the optimization problem. Note that this model can be improved by regenerating the SVR model based on new data, thus ensuring that the model is up-to-date and representative of the current situation.

For the battery model, the input vectors x contained two pieces of information: the collective input to the quadrotor and the altitude difference between the vehicle's current and desired location. The output constant y generated by this model is the vehicle's predicted flight time remaining. Over 10,000 data points were used to generate the SVR battery model using the LibSVM library in C [18]. As shown in Figure 5, the predicted flight time generated from the collective stick position and the vehicle altitude provides a reasonable estimate of the vehicle's remaining flight time. In fact, by filtering the vehicle's collective stick position and changes in altitude, a better estimate is generated. This model is currently being used by the mission planning systems in the MIT RAVEN to estimate the vehicle's remaining flight time and determine when UAVs should automatically land and recharge (as shown in Figure 6).

IV. Task Assignment with Health Constraints

The previous section illustrated how a health monitor can be implemented for vehicle components. This section will examine the task assignment problem under consideration and develop a framework for integrating health management and maintenance scheduling information into task assignment algorithms.

First, a task is defined as a tuple (w_i, p_i), where w_i is the location of task i and p_i is the priority (or value) of task i. The tasks may be known to the planning system beforehand, but more commonly, they are generated in real-time as the mission progresses. For example, during a search and track mission, new tasks are generated at the predicted future locations of the target as new information about the target velocity and position is acquired. There is a set of n vehicles $V=\{v_1,\ldots,v_n\}$ originating from a base location x_{base}, each with first-order dynamics and maximum speed $v_{\max}$. A task is called *active* if it has not been visited by a vehicle, and the set of currently active tasks is denoted by W.

Formally, the task assignment problem is to compute a mapping $T: V \rightarrow W$ which assigns a task for each vehicle to visit. The goal is to compute the map T which minimizes the total weighted service time over all the tasks:

$$\min_{T} \sum_{(w_i,p_i)\in W} p_i\ t_i$$

where t_i is the wait time before task i is performed by a vehicle. The task assignment problem in this form has received considerable attention (see [19], [20], [21], [22], [23]). However, the vehicle health state is not incorporated into these formulations. For this paper, health state information is represented by adding a fuel state to the vehicle model. In this case, the fuel model is straightforward:

- First, the vehicle's fuel level f_i decreases at a constant rate k_{fuel} anytime the vehicle is flying.
- If f_i reaches zero before the vehicle refuels, the vehicle crashes and is lost.
- In addition, the occurrence of failures is modeled as a Poisson process with time intensity ρ_f; when a failure occurs, the rate of fuel burn increases to $k_{\text{fuel,failure}} > k_{\text{fuel}}$. Thus, this failure mode increases the rate at which fuel is burned (and thus decreases the time a vehicle can complete tasks).

The following section illustrates how the Receding Horizon Task Assignment (RHTA) algorithm (defined in [24]) can be modified to include the vehicle's fuel state and how this can improve the algorithm's performance in situations where vehicles can experience failures, need refueling or require repair. The main idea - to consider the health data as information that constrains the possible list of activities each UAV can perform - can be extended to many different types of health data (engine performance, sensor performance, etc) and tasking algorithms.

Fig.6: Automated landing and recharge using the RAVEN at MIT [10]

V. RHTA Modifications

Briefly, the RHTA algorithm works as follows. Given the set of waypoints W and distances $D(i,j)$ between waypoints, RHTA enumerates all possible waypoint sequences, or *petals*, P_{vj} up to a specified length n_c. The cost of each petal is estimated as

$$S_{vp} = \sum \lambda^{T_{di}} s_{wd} \tag{3}$$

where T_{di} is the time between waypoint visits, s_{wd} are the waypoint values, and λ is a time discount factor. Given the values of all the petals S_{vp}, RHTA solves this optimization problem to select the optimal petal for each UAV:

$$\max J = \sum_{v=1}^{N_v} \sum_{p=1}^{N_{vp}} S_{vp} x_{vp} \tag{4}$$

$$\text{subj. to} \quad \sum_{v=1}^{N_v} \sum_{p=1}^{N_{vp}} A_{vpi} x_{vp} \leq 1, \quad x_{vp} \in \{0,1\} \tag{5}$$

$$\sum_{p=1}^{N_{vp}} x_{vp} = 1, \ \forall\, v \in \{1,...,N_v\} \tag{6}$$

The RHTA algorithm was extended to the health information embedded in the fuel state to the vehicle model. This was accomplished by including an estimate of each vehicle's operational radius [24], which is defined here as

$$r_i \equiv v_{\max} \frac{f_i}{k_{\text{fuel}}}$$

The quantity r_i represents the maximum distance a vehicle can fly given its current fuel state, before running out of fuel. This information can be used to effectively prune the list of petals that RHTA considers in order to ensure that the vehicle can always safely return to base before its fuel is exhausted. More specifically, the following pruning criterion was added to the RHTA algorithm (Algorithm 2.3.1 in [24]): *For every petal under consideration, reject the petal if*

$$L_i + d(w_{n_c}, x_{\text{base}}) \geq r_i$$

Here, $d(w_{n_c}, x_{\text{base}})$ represents the normal Euclidean distance between the last waypoint in the petal and the base, and

$$L_i = d(v, w_1) + \sum_{j=2}^{n_c} d(w_{j-1}, w_j)$$

is the total length of the petal. The pruning criterion rejects a petal if the length of the petal plus the distance from the terminal waypoint w_{n_c} to base is greater than the current operational radius of the vehicle. This ensures that the vehicle only visits waypoints that allow it to return safely to base.

With this extension, RHTA will assign a vehicle to return to base when *every* possible permutation of waypoints is rejected by the pruning criterion. Thus, this method provides a simple rule that determines when a vehicle should return to base for refueling since it cannot safely service any of the remaining tasks. Note that this method can create some problems if the above rule is followed too strictly since too many vehicles may be sent back to base unnecessarily (i.e. when they still have large operational radii) if there are few or no active tasks. This problem can be solved by inserting artificial *loiter tasks* $(w_{\text{loiter}}, p_{\text{loiter}})$ into W. These tasks are treated in the same way as real tasks by the RHTA algorithm, but their purpose is to force the vehicles to remain in advantageous areas.

VI. Mission-Level Maintenance Scheduling

In addition to vehicle- and task-level health monitors, mission-level health monitors can also be used improved the system's effectiveness over the duration of an operation. Monitors that evaluate mission performance, task group efficiency, team service rates and capability can be used to tune system parameters online, thereby providing a real-time evaluation of system capabilities. As shown in Figure 7, system health feedback combined with environmental data can be used to revise mission system strategies

For example, the simplified persistent surveillance mission (PSM) resource management problem can be posed as a Markov Decision Process (MDP) and formulated as an Approximate Linear Programming (ALP) problem that can be solved in real-time [25]. Here, the state $x \in S$ in the simplified three vehicle PSM problem is defined as the vector $x = (z_1, z_2, z_3, h_1, h_2, h_3)$, where z_i indicates the task to which each agent is allocated, h_i indicates each agent's maintenance/health state, and S is the state space for the problem.

Next, each action $a \in A_x$ is defined as the vector $a = (a_1, a_2, a_3)$ where a_i indicates the system's desired allocation for agent in the task space and A_x is the action space associated with the state $x \in S$. Each state x will transition under action a to the future state $y \in S$ with probability $P_a(x,y)$. Note that the agents in this problem can experience failures that cause them to be unavailable. In addition, agents are available for flight operations for a limited period of time (because of fuel, failure and maintenance concerns). Finally, a local cost for being in state x under action a is defined by the function $g_a(x) \in \mathbb{R}$.

Since this problem is posed as an MDP, an Approximate Linear Programming problem formulation can be used to

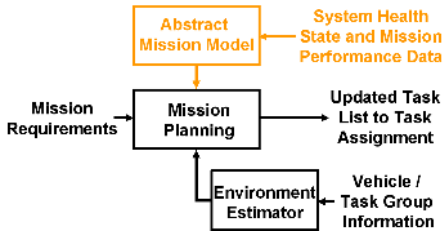

Fig.7: Updates to the mission planning subsystem in Figure 2 via health and performance data

find a feasible policy. The ALP formulation of the original Dynamic Programming problem is of the form

$$\max_{r} c^T \Phi r \tag{7}$$

$$\text{s.t.} \quad g_a(x) + \alpha \sum_{y \in S} P_a(x,y)\Phi(y)r \geq \Phi(x)r, \ \forall x \in S, \ a \in A_x$$

such that a set of M basis functions $\phi_1, \ldots \phi_M$ where $M \ll |S|$ are used to approximate the cost-to-go function $J^* \approx \Phi r^*$. Here, the approximate solution to the original problem can be used to generate a policy that directs the vehicles to meet the overall system goals, while maintaining the overall health of the mission system.

One of the major obstacles faced when using an ALP formulation of the original problem is the selection of appropriate basis vectors and state relevance weights. In many cases, selecting the appropriate parameters to find Φ is based on experience. As a result, we have developed a method allowing the autonomous mission system to automatically generate basis functions for the underlying problem to estimate the cost-to-go function that can be used in real-time [26].

VII. Results

A number of multi-vehicle tests have been flown using the RAVEN at MIT (as described in [10]) to demonstrate the mission, task assignment and control level health management algorithms. In this test suite, three UAVs equipped with cameras and 2000 mAh batteries were used to search for ground vehicles in the test area. Each vehicle-battery combination had measured flight times of 11 to 12 mins under normal operating conditions prior to these tests. The initial vehicle layout for each test is shown in Figure 8. Each UAV was placed 3 m north of the search area behind a concrete column. Therefore, these UAVs had to avoid the pole during transitions between the search and maintenance areas. In addition, two ground vehicles were used in these tests. The ground vehicle in the western end of the search area could moved via remote control. Therefore, the actions of the UAVs tracking this vehicle using vision had to be coordinated via the task advisor (running the modified version of RHTA described above) to estimate the ground object's position and velocity . Likewise, the second ground object was placed on top of a box. To accurately detect its position, UAVs had to make multiple observations of this object (from different orientations) since the terrain of the search area was not known to the mission system a priori.

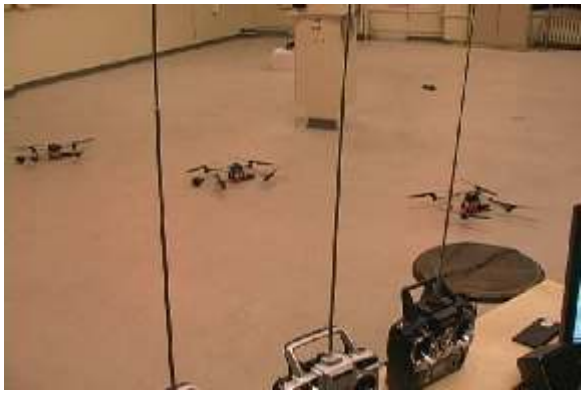

Fig.8: Multi-vehicle mission flight test setup

Flight test data from one of these test flights is shown in Figures 9 to 15. In this 12 min test flight, the camera from UAV #3 was removed to demonstrate a complete camera failure. At the beginning of this flight demonstration, the mission system commanded one vehicle to search test area for ground objects. Once airborne, the task advisor used the UAV's camera images to locate ground objects. Figure 9 shows the UAV detecting the ground objects during its search from different observation locations. Each time a ground vehicle was detected by the UAV's vision system, information of the object's location to the task advisor for future reference.

To verify the reported ground vehicle locations, the task advisor requests a second UAV from the mission manager. After the second vehicle reaches the search area, the task advisor commands both vehicles to monitor the ground vehicles. Figures 10 and 11 show UAVs #1 and #2 tracking the ground objects during the flight test. These images were generated using the RAVEN 3D operator interface playback utility. In Figure 10 both vehicles are commanded by the task advisor to remain about 90° from one another to identify the ground vehicle's location and orientation on the box. In this figure, the yellow car-like object is the actual location of the car (as determined by the RAVEN's positioning system) and the bobbin-like spherical object is the task advisor's estimate of the ground vehicle location (as generated using the processed images by the camera). Likewise, in Figure 11 both vehicles are commanded by the task advisor to stay about 90° from one another and track the moving ground object in the western end of the search area.

During the test the system's health monitors were used to manage UAVs in the flight space. As shown in Figure 12, the estimated flight time of UAV #1 was monitored by the mission system to determine when it needed maintenance during the mission. Here, the data shows that about 2.5 mins into its flight, the vehicle's battery started to degrade faster than expected. This degradation can be attributed to the fact that this vehicle was flown many times and had been involved in collision prior to this experiment. Since the motors on the vehicle were warn, UAV #1 had trouble flying with UAV #2 while tracking the moving ground vehicle (due to the fact that it got caught in UAV #2's propeller wash). Hence, 6 mins into the experiment, UAV #1 required more power to maintain its

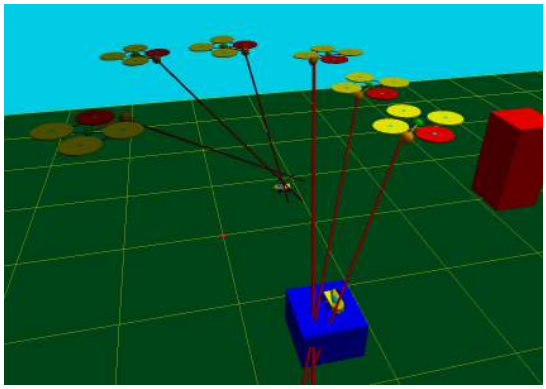

Fig.9: Single vehicle search

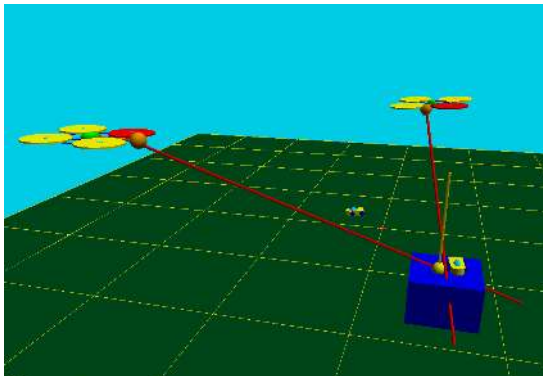

Fig.10: Two UAVs observing a ground vehicle on box

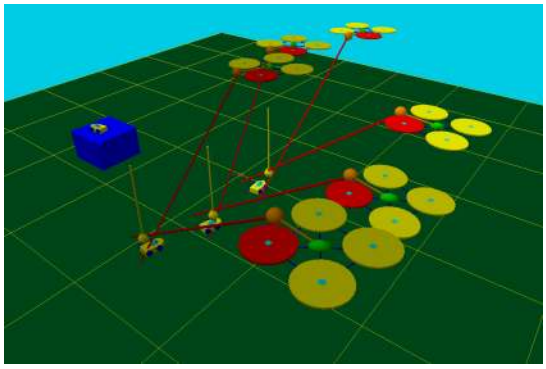

Fig.11: Two UAVs tracking a moving ground vehicle

position while observing the ground vehicles.

As shown in Figure 13, the mission system automatically commands each vehicle to participate in the mission based on the task advisor's needs and the vehicle's health. After receiving the take off command by the mission manager, vehicles required about 1 min to reach the search area. In this case, the mission system proactively commands UAV #3 to take-off (as shown in Figure 14) and replace UAV #1 since the mission system recognized that UAV #1 was using a abnormal level of collective to maintain its position during the experiment.

Normally, after the "Fuel Low" message is sent, it takes 1.5 mins for the "Fuel Warning" message to be generated for a vehicle in normal condition. However, Figure 12 shows a similar time plot of the vehicles on location in the task area. UAV #1 was commanded back to base early to ensure that it safely arrived at base to receive maintenance. To prevent

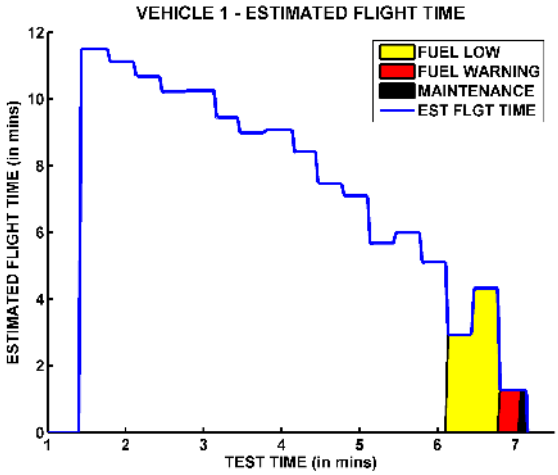

Fig.12: UAV #1 estimated vs actual flight time

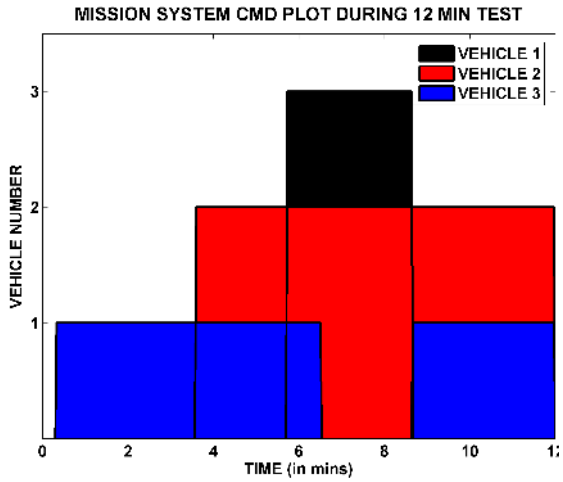

Fig.13: Mission system commands during flight

a collision, the task manager reactively modified UAV #3's flight path so that UAV #1 could safely exit the flight area, thus delaying UAV #3's entry into the search area (as shown in Figure 15).

Finally, since UAV #3 did not have a camera, the system operator manually commanded UAV #3 back to base. Since the operator was able to replace the battery on UAV #1 before failing UAV #3, the mission system recognized that a second vehicle was available. As shown in Figure 13, as soon as UAV #3 is registered as unavailable, the mission system reactively commands UAV #1 back to the search area for the rest of the test. A video of a similar multi-vehicle search test flight using vision, as well as other flights, can be found online at `http://vertol.mit.edu`.

VIII. Conclusion

In conclusion, health management techniques can be used to improve mission-level functional reliability through better system self-awareness and adaptive mission planning. The paper presents results and examples that demonstrate how health management information is being used to improve the mission system's self-awareness and adapt vehicle, guidance, task and mission plans. These algorithms, which determine

Fig.14: Two UAVs tracking a ground vehicle during vehicle cycling for maintenance

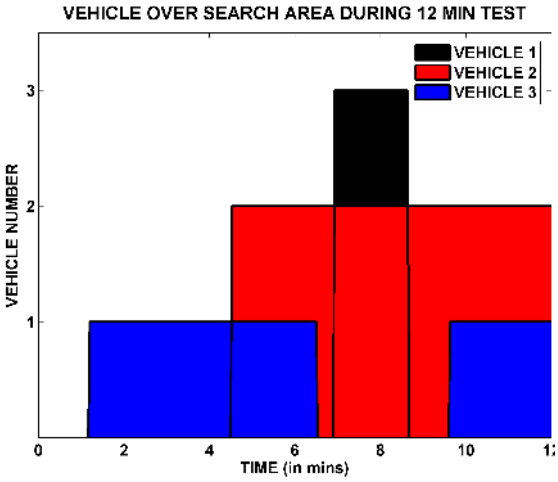

Fig.15: UAVs over the search area during test

the health of each mission component in real-time, have been successfully implemented and tested. These, and other health management algorithms for each component, are currently being used in the MIT RAVEN to improve strategic and tactical level decision making in autonomous mission systems while observing the impact of likely failures and maintenance needs for extended mission scenarios.

ACKNOWLEDGEMENTS

The authors would like to thank Spencer Ahrens, Daniel Dale and Caleb Hug for their assistance with this project. Brett Bethke would like to thank the Hertz Foundation and the American Society for Engineering Education for their support of this research. Research has been supported by the Boeing Company (Dr. John Vian at the Boeing Phantom Works) and by AFOSR grant FA9550-04-1-0458.

REFERENCES

[1] M. Fudge, T. Stagliano, and S. Tsiao, "Non-Traditional Flight Safety Systems and Integrated Vehicle Health Management Systems," ITT Industries, Advanced Engineering and Sciences Division, 2560 Alexandria Drive, Alexandria, VA 22303, Produced for the Federal Aviation Administration, August 2003.

[2] P. Gaudiano, B. Shargel, E. Bonabeau, and B. Clough, "Control of UAV SWARMS: What the Bugs Can Teach Us," in *Proceedings of the 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations Aerospace Conference*, San Diego, CA, September 2003.

[3] H. Paruanak, S. Brueckner, and J. Odell, "Swarming Coordination of Multiple UAV's for Collaborative Sensing," in *Proceedings of the 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations Aerospace Conference*, San Diego, CA, September 2003.

[4] J. Bellingham, M. Tillerson, M. Alighanbari, and J. How, "Cooperative Path Planning for Multiple UAVs in Dynamic and Uncertain Environments," in *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, NV, December 2002.

[5] J. Wohletz, D. Castanon, and M. Curry, "Closed-Loop Control for Joint Air Operations," in *Proceedings from the 2001 American Control Conference*, Arlington, VA, June 2001.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 2000.

[7] J. C. Hartman and J. Ban, "The series-parallel replacement problem," *Robotics and Computer Integrated Manufacturing*, vol. 18, pp. 215–221, 2002.

[8] D. Bertsimas and S. S. Patterson, "The Air Traffic Flow Management Problem with Enroute Capacities," *Operations Research*, vol. 46, no. 3, pp. 406–422, May-June 1998.

[9] J. Rosenberger, E. Johnson, and G. Nemhauser, "Rerouting Aircraft for Airline Recovery," *Transportation Science*, vol. 37, no. 4, pp. 408–421, November 2003.

[10] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron, "Indoor Multi-Vehicle Flight Testbed for Fault Detection, Isolation, and Recovery," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, CO, August 2006.

[11] E. King, Y. Kuwata, and J. P. How, "Experimental Demonstration of Coordinated Control for Multi-vehicle Teams," *International Journal of Systems Science*, vol. 37, no. 6, pp. 385–398, May 2006.

[12] M. Valenti, T. Schouwenaars, Y. Kuwata, E. Feron, J. How, and J. Paunicka, "Implementation of a Manned Vehicle-UAV Mission System," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Providence, RI, August 2004.

[13] Thunder Power Batteries, "Thunder Power Batteries Website," Available at http://www.thunderpower-batteries.com/, September 2006.

[14] Draganfly Innovations Inc., "Draganfly V Ti Pro Website," Available at http://www.rctoys.com/draganflyer5tipro.php, January 2006.

[15] V. Vapnik, *Statistical Learning Methods*. J. W. Wiley and Sons, 1998.

[16] S. R. Gunn, "Support Vector Machines for Classification and Regression," University of Southhampton, Tech. Rep., May 1998.

[17] A. J. Smola and B. Schölkopf, "A Tutorial on Support Vector Regression," NeuroCOLT2," Produced as part of the ESPRIT Working Group in Neural and Computational Learning II, October 1998.

[18] C.-C. Chang and C.-J. Lin, *LIBSVM: A Library for Support Vector Machines*, 2001, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[19] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson, "Coordinated Target Assignment and Intercept for Unmanned Air Vehicles," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 911–922, 2002.

[20] A. Richards, J. Bellingham, M. Tillerson, and J. How, "Coordination and Control of Multiple UAVs," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, CA, August 2002.

[21] A. E. Gil, K. M. Passino, and A. Sparks, "Cooperative Scheduling of Tasks for Networked Uninhabited Autonomous Vehicles," in *Proceedings of the 2003 IEEE Conference on Decision and Control*, Maui, HI, December 2003, pp. 522–527.

[22] C. Schumacher, P. R. Chandler, S. J. Rasmussen, and D. Walker, "Task Allocation for Wide Area Search Munitions with Variable Path Length," in *Proceedings of the 2003 American Control Conference*, Denver, CO, June 2003, pp. 3472–3477.

[23] E. Frazzoli and F. Bullo, "Decentralized Algorithms for Vehicle Routing in a Stochastic Time-Varying Environment," in *Proceedings of the 2004 IEEE Conference on Decision and Control*, 2004, pp. 3357–3363.

[24] M. Alighanbari, "Task Assignment Algorithms for Teams of UAVs in Dynamic Environments," Master's Thesis, MIT Department of Aeronautics & Astronautics, June 2004.

[25] D. P. de Farias, "The Linear Programming Approach to Approximate Dynamic Programming: Theory and Application," Ph.D. dissertation, Stanford University, June 2002.

[26] M. Valenti, "Approximate Dynamic Programming with Applications in Multi-Agent Systems," PhD Dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, June 2007.