Distributed Multi-Agent Persistent Surveillance and Tracking With Health Management

Josh Redding[*], Tuna Toksoz[†], N. Kemal Ure[*], Alborz Geramifard[*], Jonathan P. How[‡]
Aerospace Controls Laboratory, MIT

Matthew A. Vavrina[§], John Vian [¶]
Boeing Research & Technology, Seattle, WA

This paper introduces and demonstrates a full hardware testbed for research in multi-agent planning and learning for long-duration missions. The testbed includes an automated battery changing/charging platform and multiple UAV/UGV agents. The planner for each agent was formulated as a decentralised multi-agent Markov decision process and implemented using a distributed solution approach. Learning methods were also included at the agent-level to collect observations and fine-tune planning parameters for the purpose of increasing performance. Information was shared between agents according to a dynamic communication topolgy. Results are presented for persistent mission scenarios and include actuator and sensor degradations to demonstrate (1) The robustness of the mission-level planning/learning system and (2) The initially *reactive* and ultimately *proactive* qualities of the planner due to the coupling with the learning algorithm within the planning environment.

I. Introduction

In the context of teams of coordinating agents, many mission scenarios of interest are inherently long-duration and require a high level of agent autonomy due to the expense and logistical complexity of direct human control over the individual agents. Long-duration missions are practical scenarios that can show well the benefits of agent cooperation. However, such *persistent* missions can accelerate mechanical wear and tear on an agent's hardware platform, increasing the likelihood of related failures. Additionally, unpredictable failures such as the loss of a critical sensor or perhaps damage sustained during the mission may lead to severely sub-optimal mission performance. For these reasons, it is important that the planning system accounts for the possibility of such failures when constructing a mission plan. In general, planning problems that coordinate the actions

[*]Ph.D. candidate in the Aerospace Controls Lab at MIT {jredding,ure,agf}@mit.edu
[†]Research assistant in the Aerospace Controls Lab at MIT tuna@mit.edu
[‡]Richard C. Maclaurin Professor of Aeronautics and Astronautics, MIT jhow@mit.edu
[§]Research engineer at Boeing Research & Technology, Seattle, WA matthew.a.vavrina@boeing.com
[¶]J. Vian is a technical fellow at Boeing Research & Technology, Seattle, WA john.vian@boeing.com

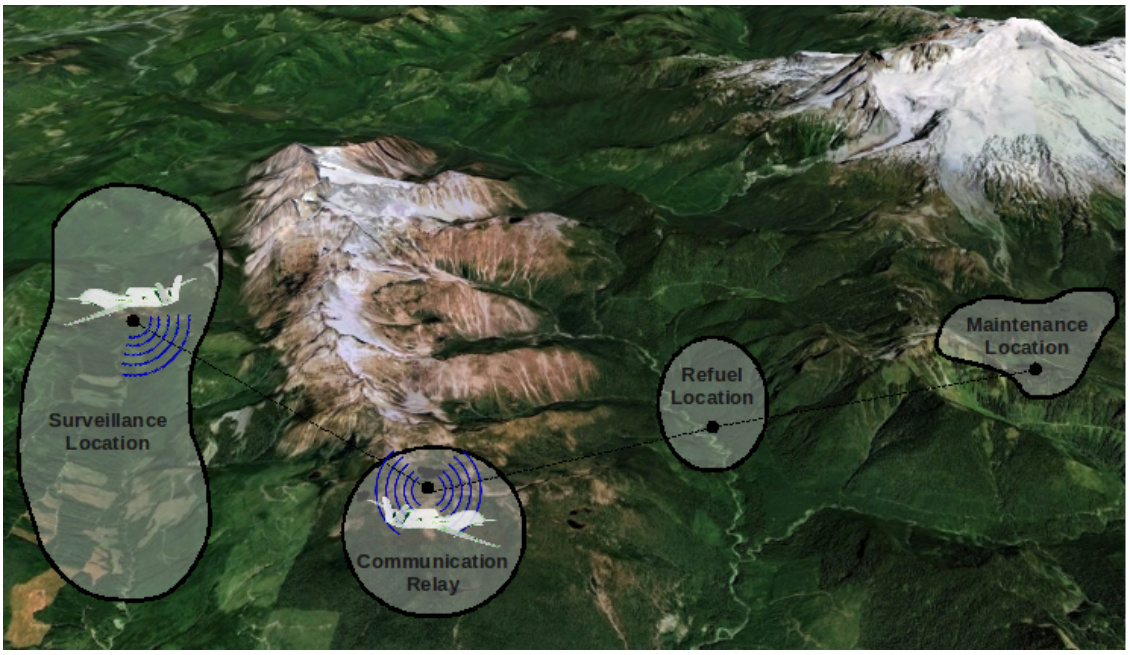

Figure 1: Mission scenario: N autonomous agents cooperate to continuously survey a specified region and to track any objects of interest discovered there. This behavior is to be maintained even under sensor and actuator health degradations.

of multiple agents, where each of which is subject to failures are referred to as *multi-agent health management* problems[1,2].

There are two typical approaches for dealing with multi-agent health management problems[3]. The first is to construct a plan based on a deterministic model of nominal agent performance. This approach ignores the possibility of failures and simply computes a new plan if failures do occur. Since the system does not anticipate the possibility of failures, but rather responds to them after they have already occurred, this approach is referred to as a *reactive* planner.

In contrast, the second approach is *proactive* and constructs a plan based on stochastic models which capture the inherent possibility of failures. Using stochastic models increases the complexity of computing the plan, as it then becomes necessary to optimize *expected* performance, where the expectation is taken over all possible scenarios that might occur. However, since proactive planners prescribe actions that mitigate the consequences of possible future failures, the resulting mission performance can be much better than that achieved by a reactive planner (naturally, depending on the validity of the underlying stochastic models). Learning methods are needed to ensure the validity of these models by tuning the associated parameters online using accumulated sensor and performance data.

This research aims to develop a complete hardware testbed to enable proactive planning and learning for teams of autonomous agents. In this paper, we implement a mission involving a group of N autonomous agents, each equipped with some type of sensor and initially situated at some base location, as shown in Figure 1. Their objectives are to continuously survey a pre-specified region of interest and to closely track any objects of interest discovered there. The problem becomes challenging when stochastic models for sensor and actuator health are included in the formulation

as well as stochastic dynamics for fuel consumption. Previous work has shown dynamic programming techniques to be a well-suited solution approach when planning for persistent missions[4]. For example, in[4] the planning problem is formulated as a Markov decision process (MDP) and solved using value iteration. The resulting optimal control policy showed a number of desirable properties, including the ability to proactively recall vehicles to base with an extra, reserve quantity of fuel which resulted in fewer vehicle losses and a higher average mission performance.

One downside to the MDP formulation is that the optimality of the resulting policy relies on the accuracy of the underlying model(s)[5,6]. If the model used to solve the MDP differs significantly from the actual system model, suboptimal performance will result when implemented on the real system. Despite efforts toward accurate models, this type of model-mismatch is frequent and, to some degree, unavoidable. This research adopts one approach that has been shown to be successful for dealing with this issue, which is to adapt the system model online and periodically re-solve the MDP using the most recent model[7].

The remainder of the paper is outlined as follows: The multi-agent persistent mission planning problem is described and formulated in Section II, following which the technical details of the learning algorithms are given in Section III. Details regarding the testbed itself are given in Section IV. Results of flight tests performed at Boeing VSTL[8] are provided in Section V, followed by some concluding remarks.

II. Problem Description

In this section, mission objectives and constraints are modeled as a Markov Decision Process (MDP). Base of this formulation is provided in the paper [3]. In this work we extend the formulation to capture the effects of recharging at the base and persistent swapping between vehicles.

II.A. Mission Description

Mission area is divided into three distinct regions geometrically. These regions are labeled as base, communication area and tasking area. Quadrotors start at the base area in landing position, take off and travel to other regions for tasking and communication duties and come back to base for recharging or repair for failed actuators and sensors. Communication area is a transition region between base and tasking areas and requires persistent existence of a quadrotor to achieve relay link in between them. This area also serves as a base for ground vehicles. Tasking area accommodates target vehicles together with a number of neutral vehicles, and is the area where persistent search and tracking takes place.

The objective of the mission is to search and detect target vehicles in the tasking area and achieve a persistent tracking once a vehicles is detected either by quadrotors or ground vehicles, while keeping a quadrotor at the communication area at all times to accomplish a relay link between base and tasking area.

There exists a number of different constraints on the mission. Each vehicle has limited fuel capacity and therefore can only operate in a limited amount of time in communication and tasking

areas. If a vehicle runs out of fuel, it goes into crashed state and cannot be recovered. A recharge station is placed in the base area to refuel the vehicles in low fuel state in order to avoid crashing. Moreover each vehicle has a probability of experiencing a sensor or actuator failure, which limits the capabilities of them which are required to perform the mission. For instance a vehicle with a failed sensor cannot perform search or tracking missions in the tasking are. All failures can be repaired once the vehicle gets back to the base.

Based on the problem description above, a reasonable plan would send quadrotors to taking area to detect the targets and pass tracking duties to ground vehicles in order to get quadrotors back to base for recharging while maintaining persistent tracking of targets. Such a plan can be described explicitly but it would not be robust to stochastic events in operation such as random actuator/sensor failures and fuel burning rates. Therefore it is desirable to model the mission as a stochastic optimal control problem which can be solved off-line to extract a reasonable policy without explicitly forcing a heuristic strategy.

In this study, mission is modeled as a finite state MDP and problem of finding the optimal policy is approached by approximate dynamic programming techniques due to fact that number of states grow exponentially with number of vehicles and discretization of the fuel state. Details on MDP formulation such as state and action space and transition dynamics are provided in the next subsection.

II.B. MDP Formulation

II.B.1. Markov Decision Processes

An infinite-horizon, discounted MDP is specified by $(\mathcal{S}, \mathcal{A}, P, g)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P_{ij}(u)$ gives the transition probability from state i to state j under action u, and $g(i,u)$ gives the cost of taking action u in state i. We assume that the MDP model is known. Future costs are discounted by a factor $0 < \alpha < 1$. A policy of the MDP is denoted by $\mu : \mathcal{S} \to \mathcal{A}$. Given the MDP specification, the problem is to minimize the cost-to-go function J_μ over the set of admissible policies Π:

$$\min_{\mu \in \Pi} J_\mu(i_0) = \min_{\mu \in \Pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \alpha^k g(i_k, \mu(i_k))\right].$$

For notational convenience, the cost and state transition functions for a fixed policy μ are defined as

$$\begin{aligned} g_i^\mu &\equiv g(i, \mu(i)) \\ P_{ij}^\mu &\equiv P_{ij}(\mu(i)), \end{aligned}$$

respectively. The cost-to-go for a fixed policy μ satisfies the Bellman equation[9]

$$J_\mu(i) = g_i^\mu + \alpha \sum_{j \in \mathcal{S}} P_{ij}^\mu J_\mu(j) \quad \forall i \in \mathcal{S}, \tag{1}$$

which can also be expressed compactly as $J_\mu = T_\mu J_\mu$, where T_μ is the (fixed-policy) dynamic programming operator.

II.B.2. State Space $\mathcal{S}$

The state of each UAV is given by two scalar variables describing the vehicle's flight status and fuel remaining. The flight status y_i describes the UAV location,

$$y_i \in \{Y_b, Y_0, Y_1, \ldots, Y_s, Y_c\}$$

where Y_b is the base location, Y_s is the surveillance location, $\{Y_0, Y_1, \ldots, Y_{s-1}\}$ are transition states between the base and surveillance locations (capturing the fact that it takes finite time to fly between the two locations), and Y_c is a special state denoting that the vehicle has crashed.

Similarly, the fuel state f_i is described by a discrete set of possible fuel quantities,

$$f_i \in \{0, \Delta f, 2\Delta f, \ldots, F_{max} - \Delta f, F_{max}\}$$

where Δf is an appropriate discrete fuel quantity. The total system state vector $\mathbf{x}$ is thus given by the states y_i and f_i for each UAV, along with r, the number of requested vehicles:

$$\mathbf{x} = (y_1, y_2, \ldots, y_n; f_1, f_2, \ldots, f_n; r)^T$$

II.B.3. Control Space $\mathcal{A}$

The controls u_i available for the i^{th} UAV depend on the UAV's current flight status y_i.

- If $y_i \in \{Y_0, \ldots, Y_s - 1\}$, then the vehicle is in the transition area and may either move away from base or toward base: $u_i \in \{\text{“}+\text{”}, \text{“}-\text{”}\}$
- If $y_i = Y_c$, then the vehicle has crashed and no action for that vehicle can be taken: $u_i = \emptyset$
- If $y_i = Y_b$, then the vehicle is at base and may either take off or remain at base: $u_i \in \{$“take off”,“remain at base”$\}$
- If $y_i = Y_s$, then the vehicle is at the surveillance location and may loiter there or move toward base: $u_i \in \{$“loiter”, “ $-\text{”}\ rbrace$

The full control vector $\mathbf{u}$ is thus given by the controls for each UAV:

$$\mathbf{u} = (u_1, \ldots, u_n)^T \tag{2}$$

II.B.4. State Transition Model P

The state transition model P captures the qualitative description of the dynamics given at the start of this section. The model can be partitioned into dynamics for each individual UAV.

The dynamics for the flight status y_i are described by the following rules:

- If $y_i \in \{Y_0, \ldots, Y_s - 1\}$, then the UAV moves one unit away from or toward base as specified by the action $u_i \in \{\text{“}+\text{”}, \text{“}-\text{”}\}$.
- If $y_i = Y_c$, then the vehicle has crashed and remains in the crashed state forever afterward.
- If $y_i = Y_b$, then the UAV remains at the base location if the action “remain at base” is selected. If the action “take off” is selected, it moves to state Y_0.
- If $y_i = Y_s$, then if the action “loiter” is selected, the UAV remains at the surveillance location. Otherwise, if the action “$-$” is selected, it moves one unit toward base.
- If at any time the UAV's fuel level f_i reaches zero, the UAV transitions to the crashed state ($y_i = Y_c$).

The dynamics for the fuel state f_i are described by the following rules:

- If $y_i = Y_b$, then f_i increases at the rate $\dot{F}_{refuel}$ (the vehicle refuels).
- If $y_i = Y_c$, then the fuel state remains the same (the vehicle is crashed).
- Otherwise, the vehicle is in a flying state and burns fuel at a stochastically modeled rate: f_i decreases by $\dot{F}_{burn}$ with probability p_{nom} and decreases by $2\dot{F}_{burn}$ with probability $(1 - p_{nom})$.

II.B.5. Cost Function g

The cost function $g(\mathbf{x}, \mathbf{u})$ penalizes three undesirable outcomes in the persistent surveillance mission. First, any gaps in surveillance coverage (i.e. times when fewer vehicles are on station in the surveillance area than were requested) are penalized with a high cost. Second, a small cost is associated with each unit of fuel used. This cost is meant to prevent the system from simply launching every UAV on hand; this approach would certainly result in good surveillance coverage but is undesirable from an efficiency standpoint. Finally, a high cost is associated with any vehicle crashes. The cost function can be expressed as

$$g(\mathbf{x}, \mathbf{u}) = C_{loc} \max\{0, (r - n_s(\mathbf{x}))\} + C_{crash} n_{crashed}(\mathbf{x}) + C_f n_f(\mathbf{x})$$

where:

- $n_s(\mathbf{x})$: number of UAVs in surveillance area in state $\mathbf{x}$,
- $n_{crashed}(\mathbf{x})$: number of crashed UAVs in state $\mathbf{x}$,
- $n_f(\mathbf{x})$: total number of fuel units burned in state $\mathbf{x}$,

and C_{loc}, C_{crash}, and C_f are the relative costs ofand loss of coverage events, crashes, and fuel usage, respectively.

II.B.6. Communications Relay Requirement

The addition of a communications relay requirement is motivated by the fact that in many UAV applications, it is necessary to maintain a communications link between the UAVs performing the mission and a fixed based location. This link may be used by human operators and/or ground-based autonomous planning systems to send commands to the UAVs, or to collect and analyze real-time sensor data from the UAVs. For example, in a search-and-rescue mission with camera-equipped UAVs, a human operator may need to observe the real-time video feeds from each UAV in order to determine probable locations of the party to be rescued. Furthermore, in many cases, the communication range of each UAV may be limited, and in particular may be less than the distance between base and the surveillance area. Therefore, in these situations, it is necessary to form a communications "chain" consisting of a spatially separated string of UAVs which relay messages back and forth between the base and the surveillance area[10,11].

In order to model the requirement for establishment of a communications chain in the MDP formulation, the cost function $g(\mathbf{x}, \mathbf{u})$ is modified in the following way. Recall that the form of the cost function is

$$g(\mathbf{x}, \mathbf{u}) = C_{loc} \max\{0, (r - n_s(\mathbf{x}))\} + C_{crash} n_{crashed}(\mathbf{x}) + C_f n_f(\mathbf{x}),$$

where $n_s(\mathbf{x})$ is a function that counts the number of UAVs in the surveillance area, and the term $C_{loc} \max\{0, (r - n_s(\mathbf{x}))\}$ serves to penalize loss of surveillance coverage (i.e. having fewer UAVs in the surveillance area than are needed). To enforce the communications requirement, let $comm(\mathbf{x})$ be a function that indicates whether communications are possible between base and the surveillance area:

$$comm(\mathbf{x}) = \begin{cases} 1 & \text{if communications link exists in state } \mathbf{x} \\ 0 & \text{otherwise} \end{cases}.$$

The functional form of $comm(\mathbf{x})$ should be chosen to reflect the communication range capabilities of each UAV. For example, if the base and surveillance locations are separated by 2 miles, and each UAV has a communication range of 1 mile, then $comm(\mathbf{x})$ should be 1 whenever there is a UAV halfway between the base and the surveillance location (since in this case, this UAV has just enough range to relay communications to both areas). In the results presented in this paper, we use a communications model of this type for $comm(\mathbf{x})$, assuming that communications are possible anytime a UAV is halfway between base and the surveillance location. Note that more complex communications models can be easily incorporated by simply changing the form of $comm(\mathbf{x})$.

Once the particular form of $comm(\mathbf{x})$ is chosen, it is incorporated into the cost function $g(\mathbf{x}, \mathbf{u})$ as follows:

$$g(\mathbf{x}, \mathbf{u}) = C_{loc} \max\{0, (r - n_s(\mathbf{x}) comm(\mathbf{x}))\} + C_{crash} n_{crashed}(\mathbf{x}) + C_f n_f(\mathbf{x}).$$

Note the only change from the original cost function is through the term $n_s(\mathbf{x}) comm(\mathbf{x})$. Thus, whenever a communications link is established, $comm(\mathbf{x})$ is 1 and the cost function behaves as

before, penalizing loss of coverage. However, if communications are broken, $comm(\mathbf{x})$ is 0 and any UAVs that are in the surveillance location become useless to the mission since they cannot communicate with base. Therefore, in order to minimize the cost $g(\mathbf{x}, \mathbf{u})$, it is necessary to maintain UAVs in the surveillance area *and* maintain a communications link, as desired.

II.B.7. Sensor Failure Model

In order to perform the surveillance missions of interest in this paper, UAVs may be equipped with a variety of sensors, such as visible-light cameras, infrared sensors, radars, etc. Of course, these sensors are not totally reliable, and in general may fail at any point during the mission. In order to develop a realistic, health-management problem formulation, it is necessary to account for the possibility of these failures in the MDP model. The qualitative description of our failure model is as follows. We assume that a UAV's sensor may fail at any point during the mission, and that the probability of failure at any given moment is described by a parameter $0 < p_{sensor\ fail} < 1$. When a sensor failure occurs, the UAV becomes useless for performing any tasks in the surveillance area. Note, however, that we assume the failure does not affect the UAV's communication subsystem, so that a UAV with a failed sensor can still perform a useful function by serving as a communications relay. Furthermore, upon returning to base, the UAV's sensor can be repaired.

In order to incorporate this failure model into the MDP, it is necessary to modify the state space $\mathcal{S}$, the system state transition model P, and the cost function $g(\mathbf{x}, \mathbf{u})$. The state space modification is straightforward; the state vector for every UAV is simply augmented with a binary variable that describes whether that UAV's sensor is failed or not. In particular, the full state vector $\mathbf{x}$ is given by

$$\mathbf{x} = (y_1, y_2, \ldots, y_n; f_1, f_2, \ldots, f_n; s_1, s_2, \ldots, s_n; r)^T,$$

where $s_i \in \{0, 1\}$ describes the sensor state of UAV i. We use the convention that $s_i = 1$ indicates that the sensor is failed, and $s_i = 0$ indicates that it is operational.

Along with the augmented state space, the transition model P must also be updated to reflect the failure dynamics of the sensors. First, whenever a UAV is flying and its sensor is already failed ($s_i = 1$), it remains failed with probability 1; if its sensor is operational ($s_i = 0$), then it fails with probability $p_{sensor\ fail}$ and remains operational with probability $(1 - p_{sensor\ fail})$. Finally, if a UAV returns to the base location ($y_i = Y_b$), then its sensor is restored to operational status ($s_i = 0$) with probability 1.

The final requirement to incorporate the sensor failure model is to update the cost function $g(\mathbf{x}, \mathbf{u})$ to reflect the fact that UAVs with failed sensor are useless for performing tasks in the surveillance area. To do this, the function $n_s(\mathbf{x})$, which previously counted the total number of UAVs in the surveillance area, is modified to instead count the number of UAVs in the surveillance area *with operational sensors.*

II.B.8. Recharge Requirement

Formulation above allows vehicles to takeoff from landing area without fully recharging, since recharge station has a constant recharging rate and unlike repair of sensor and actuator failures, a vehicle is not instantaneously switched to fully charged state once it arrives at the base. In order to avoid this situation we place the following constraint on system dynamics:

- If $y_i = Y_b$, then the vehicle is at base and may remain at base: $u_i \in \{\text{“remain at base”}\}$ or it may take-off if it is fully charged $y_i = Y_b$ and $f_i = F_{max}$ then $u_i \in \{\text{“remain at base”},\text{“take-off”}\}$

Therefore vehicle can leave the base if and only if it is fully charged.

II.B.9. Persistent Swapping Requirement

Since transition dynamics evolve in discrete time in the current formulation, swapping of two vehicles between communication and tasking areas assumed to be occurred instantaneously which results in loss of coverage of tasking area during swapping. A reasonable design choice would be penalizing the situation where a vehicle leaves the tasking area without any backup vehicle to replace it. For this purpose we count the number of vehicles which are commanded to leave the task area and commanded to stay at the task area , represented by $n_{leaving}(\mathbf{x},\mathbf{u})$ and $n_{staying}(\mathbf{x},\mathbf{u})$ respectively. Then cost function is updated into its final form where C_{swap} is the cost of swapping:

$$\begin{aligned} g(\mathbf{x},\mathbf{u}) \;=\; & C_{loc}\max\{0,(r - n_s(\mathbf{x})comm(\mathbf{x}))\} + \\ & C_{swap}\max\{0,(n_{leaving}(\mathbf{x},\mathbf{u}) - n_{staying}(\mathbf{x},\mathbf{u})comm(\mathbf{x}))\} + C_{crash} \\ & n_{crashed}(\mathbf{x}) + C_f n_f(\mathbf{x}). \end{aligned}$$

With latest addition to cost function, situations where no penalty occurs corresponds to state where the number of vehicles leaving the task area is equal to the number of vehicles staying in the task area, such that leaving the tasking area does not have a negative effect on coverage.

III. Multi-Agent Learning

Planning for teams of heterogeneous autonomous mobile agents in stochastic systems is a challenging problem arising in many domains such as robotics, aviation, and military applications. While cooperative planners provide fast and reliable solutions to these problems [12–20], their solutions are sub-optimal due to the inaccuracy of the model (*e.g.,* the exact model is approximated by a linear system), or violation of assumptions (*e.g.,* the experienced noise does not obey a Gaussian disribution). Moreover, the output of cooperative planners are often nominal trajectories corresponding to a narrow part of the state space. The limited applicability of the solution requires recalculation of the plan after each deviation from the nominal path. By relaxing most assumptions made by cooperative planners, online reinforcement learning (RL) techniques operate in a

more realistic framework where the system reasons about the consequences of its own actions and improves its future performance without the need of a third party. Additionally, RL techniques provide global policies executable from anywhere in the state space. However, applying RL methods to multi-agent domains involves three main challenges:

(I) Sample Complexity: For multi-agent domains, the size of the state space is often very large as it is exponential in the number of agents. For example a domain with 10 agents where each agent can take 2 modes and 50 positions has 10^{20} possible states. On the other hand, the number of interactions required by RL methods to achieve reasonable performance often grows with the size of the state space. Hence, scaling RL methods to multi-agent domains is challenging.

(II) Limited Computation: Onboard computation is typically limited, particularly on airborn platforms. In addition, embedding learning within the normal operation of the system (*i.e.,* online setting) results in a very limited amount of time allowed for learning on each interaction.

(III) Safe Exploration: While catastrophic outcomes provide learners with valuable information, sometime the loss involved in perceiving such feedbacks is not sustainable. For example, crashing a fuel-limited unmanned aerial vehicle (UAV), while carrying the message that running out of fuel is bad, costs a lot of money and potentially endangers the life of nearby civilians. Avoiding such fatal states requires a robust learning scheme where the range of exploration is bounded within some safe region.

III.A. Temporal Difference Learning

The temporal difference (TD) error after taking action a_t from state s_t at time t is defined as the difference between the current value for the state-action pair and the estimated Q value based on the observed reward received after action a_t and the value for the next state-action pair in the trajectory:

$$\delta_t(Q^\pi) = r_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t). \tag{3}$$

TD methods[21] use the TD error at each time step as a gradient for reducing the error in the Q function estimation.

III.B. Linear Function Approximation

A lookup-table representation of the Q function is impractical when the state space is large, and a common approach is to use a linear function approximation of the form $Q^\pi(s,a) = \theta^T \phi(s,a)$, where $\phi : \mathcal{S} \times \mathcal{A} \to \Re^n$ is the basis function and θ is a weight vector. Each element of the basis function $\phi(s)$ is called a *feature* ; $\phi_f(s) = c$ denotes feature f is c in state s.[a] The representation for Q function is formed by the space spanned by the linear combination of basis functions of all states, and hence named the feature representation. Since adaptive function approximation is central to

[a]For readability, $\phi(s)$ is used instead of $\phi(s,a)$, but ϕ is always conditioned on the action implicitly.

this document, let $\phi_f^t(s)$ indicate the value of feature f corresponding to the basis function at time t for state s. Given a feature representation formed by ϕ, several methods[22,23] exist to learn the weight vector θ.

Of special interest is the set of basis functions ϕ where the output is a binary vector ($\phi : \mathcal{S} \times \mathcal{A} \rightarrow \{0,1\}^n$); not only can the corresponding weights θ be computed efficiently[24], but the weights also indicate the importance of each binary property[25]. If ϕ contains a unique non-zero feature for each state-action pair, then the function approximation reduces to a lookup table that assigns a value to each state-action pair.

Value based RL techniques, a popular family of RL, calculate the optimal control policy by estimating the long term advantage of each action from every state and then acting greedily with respect to those values. Function approximators have scaled RL methods to large domains by reducing their sample complexity[26,27], allowing RL methods to exceed the human level of expertise in several domains[25,28,29]. In particular, practitioners have favored the linear family of approximators[26,30–34] due to their desirable properties such as theoretical analysis[35] and cheap computational complexity[36].

A fundamental open problem in the realm of RL is how to pick basis functions for the linear function approximator in order to make the task learnable. While for most applications, the domain expert selects the set of basis functions through manual tuning[25], in recent years a substantial body of research has been dedicated towards automating this process resulted in adaptive function approximators (AFAs)[31,32,34,37–40]. Finding a computationally cheap AFA which scales to large domains, provides convergence guarantees when combined with RL techniques, and requires minimal design skill is still an open problem.

Safe exploration is another hurdle that practitioners come across when applying RL to realistic domains. Existing methods for safe exploration either behave too pessimistically[41], lack convergence guarantees[42], or provide no safety guarantees[43]. This thesis introduces cooperative learning methods that take advantage of existing cooperative planners by regarding them as safe policies in order to verify the safety of the action suggested by the learner. Furthermore, solutions generated by cooperative planners will guide the learning exploration regime to focus on promising parts of the state space, thus addressing challenges (I) and (III).

IV. Experimental Setup

To demonstrate the capability of problem formulation and the optimal solution extracted from it, our algorithms were implemented on an experimental testbed centered on the Boeing Vehicle Swarm Technology Laboratory (VSTL). A custom designed recharge station and a group of UAV/UGV were integrated into test platform for this purpose. This section provides details on each component of the testbed in terms of hardware specifications.

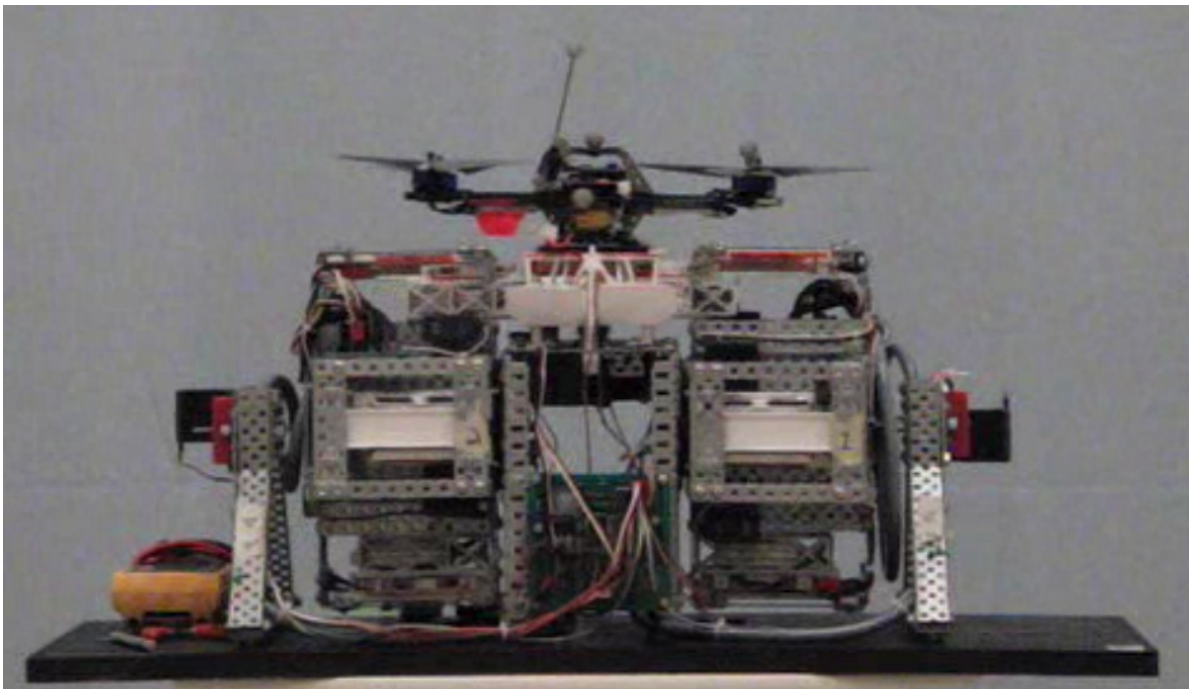

Figure 2: Recharge station

IV.A. Recharge Station

One of the purposes of this paper is to introduce and demonstrate a complete testbed for multi-agent planning and learning over extended-duration missions. As missions involving unmanned aerial vehicles (UAVs) stretch in duration, frequent and robust refueling becomes critical to overall success. In the research setting, refueling typically means charging or swapping batteries onboard a small-scale UAV. Automating and/or streamlining this procedure has been the topic of much previous work[44]? ? ? ? ? . However, waiting for a battery to properly charge can be time-consuming, adding undesirable delays in the overall mission. Also, "cold" battery-swapping techniques require a complete shutdown of the vehicle's onboard electronics as the battery is swapped for a new one, losing onboard data and state information at each recharge cycle. As slow recharge times and cold battery swaps are undesirable, this research introduces an automated, portable landing platform capable of "hot" swapping batteries (*e.g.,* swapping batteries in a manner such that the UAV never loses power in the process) then recharging them offline. In addition, as shown in Figure 2, the automated station holds a buffer of 8 batteries, all of which are continuously charging, and uses a novel dual-drum structure to automatically select which battery to swap into the UAV. The hot-swap capability, in combination with local recharging and a large battery capacity allow this platform to refuel multiple UAVs for long-duration and persistent missions without incurring major delays or vehicle shutdowns. A fully-functional hardware version of the automated recharge station is tested in the context of a multi-agent, long-duration persistent mission where surveillance is continuously required over a specified region[45]. In this context, the recharge station provides a fast, reliable refueling capability that allows for persistent coverage of the surveiallance region and

the success of the mission.

IV.B. Agent Platforms

Four different agent platforms were used in this research consisting of a quadrotor UAV and three variations of ground-based agents. Figure 3 shows these agents with the UAV in the upper-left, the team member UGV in the upper-right, the target UGV in the lower-left and the civilian UGV in the lower-right.

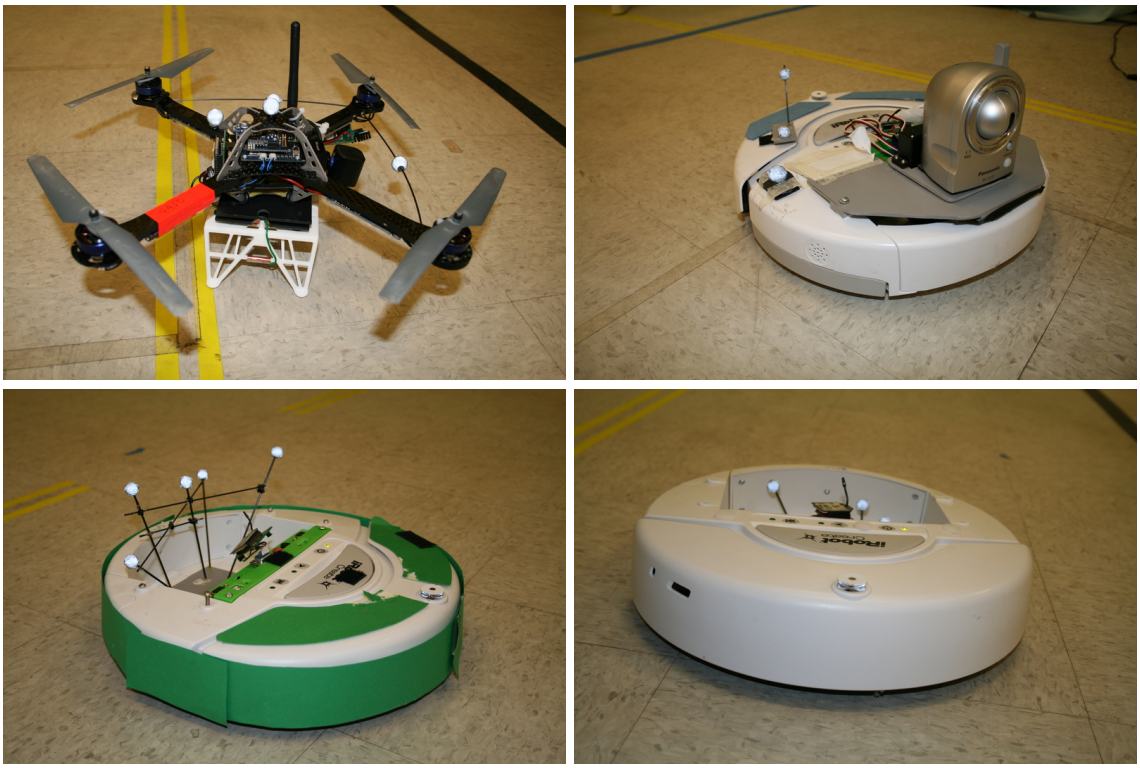

Figure 3: Agent platforms

As seen in Figure 3, the quadrotor UAVs were equipped with a wifi-enabled webcam and a recharge-capable base. Team member UGVs were also equipped with onboard cameras to detect the target UGVs amongst the "civilians".

IV.C. Test Facility

Boeing Research and Technology has developed the Vehicle Swarm Technology Laboratory (VSTL), an environment for testing a variety of vehicles in an indoor, controlled environment[8]. VSTL is capable of simultaneously supporting a large number of both air and ground vehicles, thus providing a significant advantage over traditional flight test methods in terms of flight hours logged. As seen in Figure 4, the primary components of the VSTL are:

- A camera-based motion capture system for reference positions, velocities, attitudes and attitude rates

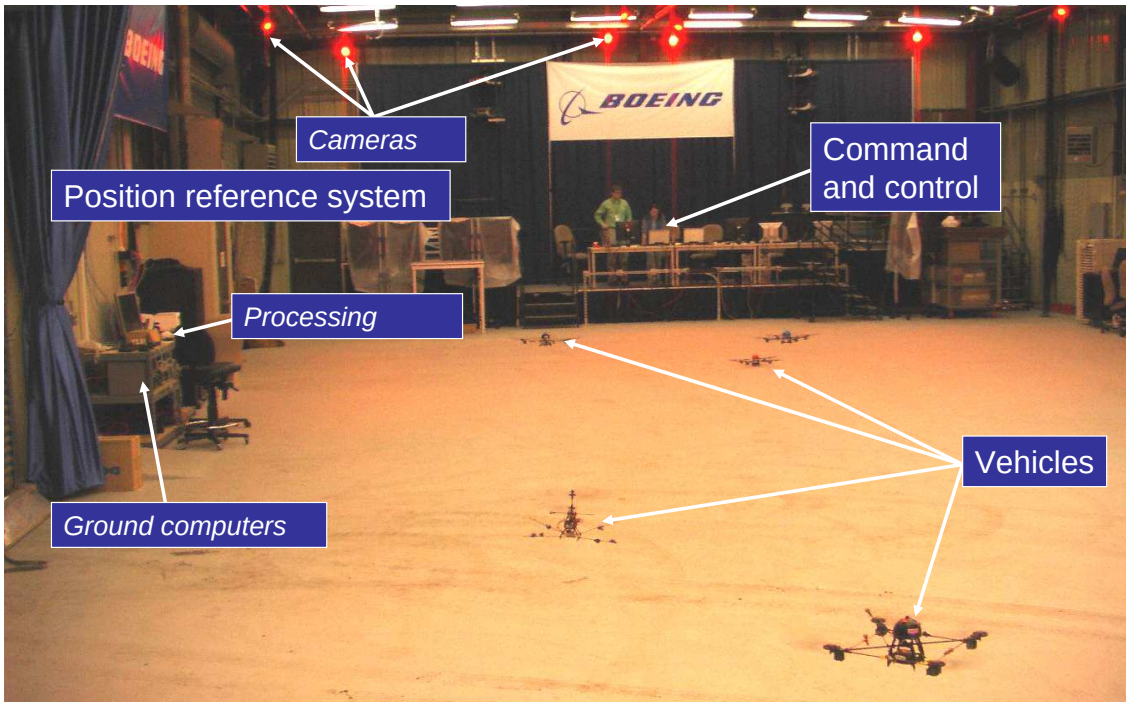

Figure 4: The Boeing Vehicle Swarm Technology Laboratory (VSTL), a state-of-the-art rapid prototyping indoor flight testing facility[8].

- A cluster of off-board computers for processing the reference data and calculating control inputs
- Operator interface software for providing high-level commands to individual and/or teams of agents

These components are networked within a systematic, modular architecture to support rapid development and prototyping of multi-agent algorithms[8].

V. Results

Figure 5 shows a sequence of snapshots taken directly from an implementation of the mission described in Section II. Moving from the upper-left to the lower-right image in Figure 5, we see agents: leaving the base area to begin the persistent surveillance task, searching the area for targets, tracking a discovered target, and returning to base for a fresh battery.

Forthcoming: There will be a plot showing robustness to failures in sensors and actuators. There will be a plot showing the convergence of the learning method over the course of the mission. There will be a snapshot of fuel vs time for an agent, showing that the recharge station indeed enables long-duration missions.

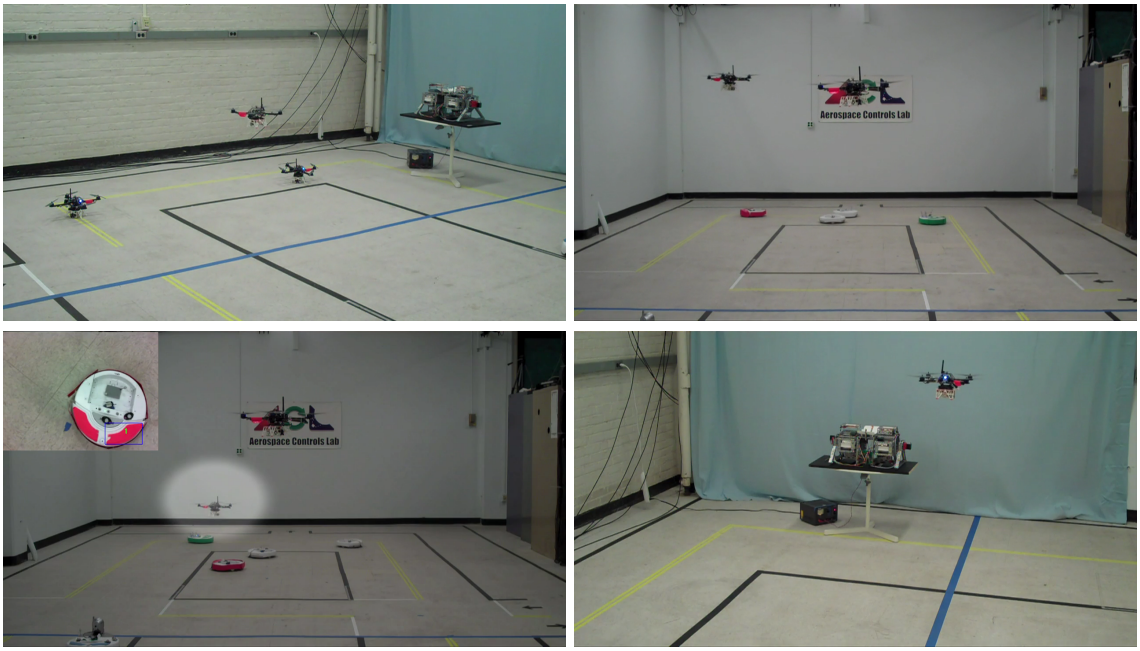

Figure 5: Mission snapshots

VI. Conclusion

In conclusion, this paper has introduced and demonstrated a full hardware testbed for research in multi-agent planning and learning for long-duration missions. The testbed includes an automated battery changing/charging platform and multiple UAV/UGV agents. The planner for each agent was formulated as a decentralized multi-agent Markov decision process and implemented using a distributed solution approach. Learning methods were also included at the agent-level to collect observations and fine-tune planning parameters for the purpose of increasing performance.

Acknowledgments

This research was supported by Boeing Research & Technology and in part by AFOSR (FA9550-09-1-0522).

References

[1] Valenti, M., Bethke, B., How, J. P., de Farias, D. P., and Vian, J., "Embedding Health Management into Mission Tasking for UAV Teams," *American Control Conference (ACC)*, 9-13 July 2007, pp. 5777–5783.

[2] Valenti, M., Dale, D., How, J., and Vian, J., "Mission Health Management for 24/7 Persistent Surveillance Operations," *AIAA Guidance, Navigation, and Control Conference (GNC)*, Myrtle Beach, SC, August 2007.

[3] Bethke, B., How, J. P., and Vian, J., "Multi-UAV Persistent Surveillance With Communication

Constraints and Health Management," *Guidance Navigation and Control Conference*, August 2009.

[4] Bethke, B., How, J., and Vian, J., "Group Health Management of UAV Teams With Applications to Persistent Surveillance," *American Control Conference (ACC)*, 2008.

[5] Iyengar, G., "Robust Dynamic Programming," *Math. Oper. Res.*, Vol. 30, No. 2, 2005, pp. 257–280.

[6] Nilim, A. and Ghaoui, L. E., "Robust Control of Markov Decision Processes with Uncertain Transition Matrices," *Operations Research*, Vol. 53, No. 5, Sept.-Oct. 2005, pp. 780–798.

[7] Bethke, B., Bertuccelli, L. F., and How, J. P., "Experimental Demonstration of Adaptive MDP-Based Planning with Model Uncertainty." *AIAA Guidance Navigation and Control*, Honolulu, Hawaii, 2008.

[8] Saad, E., Vian, J., Clark, G., and Bieniawski, S., "Vehicle Swarm Rapid Prototyping Testbed," *AIAA Infotech@Aerospace*, Seattle, WA, 2009.

[9] Bertsekas, D., *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, MA, 2007.

[10] Dixon, C. and Frew, E., "Decentralized Extremum-Seeking Control of Nonholonomic Vehicles to Form a Communication Chain," *Lecture notes in Control and Information Sciences*, Vol. 369, 2007, pp. 311.

[11] Dixon, C. and Frew, E., "Maintaining optimal communication chains in robotic sensor networks using mobility control," *Proceedings of the 1st international conference on Robot communication and coordination*, IEEE Press Piscataway, NJ, USA, 2007.

[12] Casal, A., *Reconfiguration Planning for Modular Self-Reconfigurable Robots*, Ph.D. thesis, Stanford University, Stanford, CA, 2002.

[13] Beard, R., McLain, T., Goodrich, M., and Anderson, E., "Coordinated Target Assignment and Intercept for Unmanned Air Vehicles," *IEEE Transactions on Robotics and Automation*, Vol. 18(6), 2002, pp. 911–922.

[14] Xu, L. and Ozguner, U., "Battle management for unmanned aerial vehicles," *IEEE Conference on Decision and Control*, Vol. 4, 9-12 Dec. 2003, pp. 3585–3590 vol.4.

[15] Alighanbari, M., Kuwata, Y., and How, J. P., "Coordination and control of multiple UAVs with timing constraints and loitering," *American Control Conference (ACC)*, Vol. 6, 4-6 June 2003, pp. 5311–5316 vol.6.

[16] Ryan, A., Zennaro, M., Howell, A., Sengupta, R., and Hedrick, J. K., "An overview of emerging results in cooperative UAV control," *IEEE Conference on Decision and Control*, 2004, pp. 602–607.

[17] Alighanbari, M., *Task assignment algorithms for teams of UAVs in dynamic environments*, Master's thesis, Massachusetts Institute of Technology, 2004.

[18] Saligrama, V. and Castañón, D., "Reliable Distributed Estimation with Intermittent Communications," *IEEE Conference on Decision and Control*, Dec. 2006, pp. 6763–6768.

[19] Wang, X., Yadav, V., and Balakrishnan, S. N., "Cooperative UAV Formation Flying With

Obstacle/Collision Avoidance," *Control Systems Technology, IEEE Transactions on*, Vol. 15, No. 4, 2007, pp. 672–679.

[20] Choi, H., Brunet, L., and How, J., "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. on Robotics*, Vol. 25, No. 4, 2009, pp. 912 – 926.

[21] Sutton, R. S., "Learning to Predict by the Methods of Temporal Differences," *Machine Learning*, Vol. 3, 1988, pp. 9–44.

[22] Sutton, R. S. and Barto, A. G., *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[23] Lagoudakis, M. G. and Parr, R., "Least-Squares Policy Iteration," *Journal of Machine Learning Research*, Vol. 4, 2003, pp. 1107–1149.

[24] Buro, M., "From Simple Features to Sophisticated Evaluation Functions," *Computers and Games, Proceedings of CG98, LNCS 1558*, Springer-Verlag, 1999, pp. 126–145.

[25] Silver, D., Sutton, R. S., and Müller, M., "Sample-based learning and search with permanent and transient memories," *ICML '08: Proceedings of the 25th international conference on Machine learning*, ACM, New York, NY, USA, 2008, pp. 968–975.

[26] Sutton, R. S., "Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding," *Advances in Neural Information Processing Systems 8*, The MIT Press, 1996, pp. 1038–1044.

[27] Stone, P., Sutton, R. S., and Kuhlmann, G., "Reinforcement Learning for RoboCup Soccer Keepaway," *International Society for Adaptive Behavior*, Vol. 13, No. 3, 2005, pp. 165–188.

[28] Zhang, W. and Dietterich, T. G., "High-Performance Job-Shop Scheduling With A Time-Delay TD(lambda) Network," *Advances in Neural Information Processing Systems 8*, MIT Press, 1995, pp. 1024–1030.

[29] Tesauro, G., "Programming backgammon using self-teaching neural nets," *Artificial Intelligence*, Vol. 134, No. 1-2, 2002, pp. 181–199.

[30] Bowling, M. and Veloso, M., "Simultaneous adversarial multirobot learning," 2003.

[31] Ratitch, B. and Precup, D., "Sparse Distributed Memories for On-Line Value-Based Reinforcement Learning," *ECML*, edited by J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, Vol. 3201 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 347–358.

[32] Waldock, A. and Carse, B., "Fuzzy Q-Learning with an adaptive representation," *Fuzzy Systems, IEEE World Congress on Computational Intelligence*, 2008, pp. 720–725.

[33] Li, L., Williams, J. D., and Balakrishnan, S., "Reinforcement Learning for Dialog Management using Least-Squares Policy Iteration and Fast Feature Selection," *New York Academy of Sciences Symposium on Machine Learning*, 2009.

[34] Petrik, M., Taylor, G., Parr, R., and Zilberstein, S., "Feature Selection Using Regularization in Approximate Linear Programs for Markov Decision Processes," *ICML '10: Proceedings of the 27th Annual International Conference on Machine Learning*, 2010.

[35] Tsitsiklis, J. N. and Van Roy, B., "An Analysis of Temporal-Difference Learning with Function Approximation," *IEEE Transactions on Automatic Control*, Vol. 42, No. 5, 1997, pp. 674–690.

[36] Geramifard, A., Bowling, M., and Sutton, R. S., "Incremental Least-Square Temporal Difference

Learning," *The Twenty-first National Conference on Artificial Intelligence (AAAI)*, 2006, pp. 356–361.

[37] Reynolds, S. I., "Adaptive Resolution Model-Free Reinforcement Learning: Decision Boundary Partitioning," *In Proc. 17th International Conf. on Machine Learning*, Morgan Kaufmann, 2000, pp. 783–790.

[38] Sherstov, A. A. and Stone, P., "Function Approximation via Tile Coding: Automating Parameter Choice," *Symposium on Abstraction, Reformulation, and Approximation (SARA)*, edited by J.-D. Zucker and I. Saitta, Vol. 3607 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, Berlin, 2005, pp. 194–205.

[39] Whiteson, S., Taylor, M. E., and Stone, P., "Adaptive Tile Coding for Value Function Approximation," Tech. Rep. AI-TR-07-339, University of Texas at Austin, 2007.

[40] Lin, S. and Wright, R., "Evolutionary Tile Coding: An Automated State Abstraction Algorithm for Reinforcement Learning," *AAAI Workshops*, 2010.

[41] Heger, M., "Consideration of Risk and Reinforcement Learning," *Machine Learning, Proceedings of the 11th International Conference*, 1994, pp. 105–111.

[42] Geibel, P. and Wysotzki, F., "Risk-sensitive reinforcement learning applied to chance constrained control," *JAIR*, Vol. 24, 2005.

[43] Abbeel, P. and Ng, A. Y., "Exploration and apprenticeship learning in reinforcement learning," *in Proc. 21st International Conference on Machine Learning*, ICML, 2005, pp. 1–8.

[44] Valenti, M., Bethke, B., How, J., de Farias, D., and Vian, J., "Embedding Health Management into Mission Tasking for UAV Teams," *American Control Conference (ACC)*, 9–13 July 2007, pp. 5777–5783.

[45] Bethke, B., How, J., and Vian, J., "Multi-UAV Persistent Surveillance With Communication Constraints and Health Management," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Chicago, IL, August 2009.