Agent capability in persistent mission planning using approximate dynamic programming

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

Citation

As Published

Publisher
Institute of Electrical and Electronics Engineers

Version
Final published version

Citable link
http://hdl.handle.net/1721.1/58888

Terms of Use
Article is made available in accordance with the publisher’s policy and may be subject to US copyright law. Please refer to the publisher’s site for terms of use.
Agent Capability in Persistent Mission Planning using Approximate Dynamic Programming

Brett Bethke, Joshua Redding and Jonathan P. How
Matthew A. Vavrina and John Vian

Abstract—This paper presents an extension of our previous work on the persistent surveillance problem. An extended problem formulation incorporates real-time changes in agent capabilities as estimated by an onboard health monitoring system in addition to the existing communication constraints, stochastic sensor failure and fuel flow models, and the basic constraints of providing surveillance coverage using a team of autonomous agents. An approximate policy for the persistent surveillance problem is computed using a parallel, distributed implementation of the approximate dynamic programming algorithm known as Bellman Residual Elimination. This paper also presents flight test results which demonstrate that this approximate policy correctly coordinates the team to simultaneously provide reliable surveillance coverage and a communications link for the duration of the mission and appropriately retasks agents to maintain these services in the event of agent capability degradation.

I. INTRODUCTION

In the context of multiple 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 individual agents. Hence, autonomous mission planning and control for multi-agent systems is an active area of research [1], [2].

Long-duration missions are indeed 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 sub-optimal mission performance. For these reasons, it is important that the planning system accounts for the possibility of such failures when devising a mission plan. In general, planning problems that coordinate the actions of multiple agents, where each of which is subject to failures are referred to as multi-agent health management problems [3], [4].

In general, there are two approaches to dealing with the uncertainties associated with possible agent failure modes [5]. The first is to construct a plan based on a deterministic model of nominal agent performance (in effect, this approach simply ignores the possibility of failures). Then, if failures do occur, a new plan is computed in response to the failure. 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 to construct a plan based on a stochastic model which captures the inherent possibility of failures. Using a stochastic model increases the complexity of computing the plan, since it may be necessary to optimize a measure of expected performance, where the expectation is taken over all possible scenarios that might occur. However, since this proactive planning approach can anticipate the possibility of failure occurring in the future and take actions to mitigate the consequences, the resulting mission performance may be much better than the performance that can be achieved with a reactive planner. For example, if a UAV tracking a high value target is known to have a high probability of failure in the future, a proactive planner may assign a backup UAV to also track the target, thus ensuring un-interrupted tracking if one of the vehicles were to fail. In contrast, a reactive planner would only have sent a replacement for an already-failed UAV, likely resulting in the loss of the tracked target, at least for some period of time.

In Ref [6], the use of dynamic programming techniques is motivated when formulating and solving a persistent surveillance planning problem. In this problem, there is a group of UAVs equipped with cameras or other types of sensors. The UAVs are initially located at a base location, which is separated by a large distance distance from the surveillance location. The objective of the problem is to maintain a specified number of UAVs over the surveillance location...
at all times. Furthermore, the fuel usage dynamics of the
UAVs are stochastic, requiring the use of health management
techniques to achieve good surveillance coverage. In the
approach taken in [6], the planning problem is formulated
as a Markov Decision Process (MDP) [7] and solved using
value iteration. The resulting optimal control policy for
the persistent surveillance problem exhibits a number of
desirable properties, including the ability to proactively recall
vehicles to base with an extra, reserve quantity of fuel,
resulting in fewer vehicle losses and a higher average mission
performance. The persistent surveillance MDP formulation
relies on an accurate estimate of the system model (i.e.
state transition probabilities) in order to guarantee optimal
behavior [8], [9]. If the model used to solve the MDP differs
significantly from the true system model, suboptimal perfor-
ance will result. This section gives the specific formulation of the persistent
mission MDP from the formation of the problem state space
in section II-A.1, to the cost function in section II-A.4.

A. Persistent Mission Problem

This section gives the specific formulation of the persistent
mission MDP from the formation of the problem state space
in section II-A.1, to the cost function in section II-A.4.

1) State Space \( S \): The state of each UAV is given by three
scalar variables describing the vehicle’s flight status, fuel
remaining and sensor status. The flight status \( y_i \) describes
the UAV location,

\[
y_i \in \{Y_0, Y_1, \ldots, Y_n, Y_c\}
\]

where \( Y_0 \) 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 \( \Delta f \) is an appropriate discrete fuel quantity.

The sensor state \( s_i \) is described by a discrete set of
sensor attributes, \( s_i \in \{0, 1, 2\} \) representing each sensor as
functional, impaired and disabled respectively.

The total system state vector \( x \) is thus given by the states
\( y_i, f_i \) and \( s_i \) for each UAV, along with \( r \), the number of
requested vehicles:

\[
x = (y_1, y_2, \ldots, y_n; f_1, f_2, \ldots, f_n; s_1, s_2, \ldots, s_n; r)^T
\]

2) Control Space \( A \): 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: \( u_i \in \{+n, -n\} \)
- 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: \( u_i \in \{ \text{"take off"}, \text{"remain at base"} \} \).

If \( y_i = Y_s \), then the vehicle is at the surveillance location and may loiter there or move toward base: \( u_i \in \{ \text{"loiter"}, \text{"-"} \} \).

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

\[
\mathbf{u} = (u_1, \ldots, u_n)^T
\]  

3) 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 or toward base as specified by the action \( u_i \in \{ \text{"+"}, \text{"-"} \} \).
- If \( y_i = Y_c \), then the vehicle has crashed and remains in the crashed state forever afterward.
- If \( y_i = Y_b \) and 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: \( \dot{f}_i \) decreases by \( \dot{f}_{burn} \) with probability \( p_{nom} \) and decreases by \( 2\dot{f}_{burn} \) with probability \( 1 - p_{nom} \).

4) Cost Function \( g \): The cost function \( g(x, u) \) penalizes several undesirable outcomes in the persistent surveillance mission. First, any gaps in surveillance coverage (i.e., times when fewer vehicles are in the surveillance area than were requested) are penalized with a high cost. Second, a small penalty is incurred for vehicles in the surveillance area with inadequate sensor capability. An additional small cost is associated with each unit of fuel used, which prevents 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

\[
g(x, u) = C_{loc}\delta_n + C_{ds}n_{ds}(x) + C_cn_c(x) + C_fn_f(x)
\]

where:

- \( \delta_n = \max\{0, (r - n_s(x))\} \)
- \( n_s(x) \): number of UAVs in surveillance area in state \( x \),
- \( n_{ds}(x) \): number of UAVs in surveillance area with degraded sensor in state \( x \),
- \( n_c(x) \): number of crashed UAVs in state \( x \),
- \( n_f(x) \): total number of fuel units burned in state \( x \),

and \( C_{loc}, C_{ds}, C_c, \text{ and } C_f \) are respectively the relative costs of loss of coverage events, vehicles in the surveillance location with a degraded sensor, crashes, and fuel usage.

5) Approximate Solutions to Dynamic Programming: Bellman Residual Elimination (BRE) is a approximate policy iteration technique that is closely related to the class of Bellman residual methods [16]-[18]. Bellman residual methods attempt to perform policy evaluation by minimizing an objective function of the form

\[
\left( \sum_{i \in S} |\tilde{J}_\mu(i) - T_\mu\tilde{J}_\mu(i)|^2 \right)^{1/2}
\]

where \( \tilde{J}_\mu \) is an approximation to the true cost function \( J_\mu \) and \( S \subset S \) is a set of representative sample states. BRE uses a flexible kernel-based cost approximation architecture to
construct \( \hat{J}_\mu \) such that the objective function given by Eq. (6) is identically zero. Details outlining the implementation of the BRE algorithm can be found in [5]. As implemented, the BRE solver runs continuously, being updated with agent states in real-time and queried for a policy as needed.

### III. Agent Health Monitoring & Capability Assessment

An onboard health monitoring system collects local data and provides some measure of agent health. The contents of this measure can vary greatly, depending on the application domain. In this research, the onboard health monitor collects available state information and provides a metric indicating the health, or the deviation of actual from nominal efficiency, of each actuator, including onboard sensors. This provides key insight into the capability of the mobile agent, and its ability to perform its advertised range of tasks.

#### A. Capability Assessment

The information provided by a health monitor can be very detailed. However, for the sake of communication and planner efficiency, it is desirable to abstract the detailed information down to a sufficient statistic or minimal covering set of parameters and pass these up from the agent- to the system-level. For example, it is not difficult to imagine that a quadrotor vehicle with a failing left/right motor will not be able to perform intense rolling or yawing maneuvers nominally well. The question then becomes, what level of detail relating to agent health do we wish to pass along to the planner? The abstraction process used in this research represents an obvious first choice: a single scalar value, roughly encapsulating the vehicle capability with respect to nominal actuator performance.

In this formulation of the agent capability assessment module, three possible capability levels are defined based on observed quadrotor performance with a single degraded motor. A quadrotor flying with motors/sensors operating nominally is assigned a capability value of 0 and can execute any tracking or surveillance task or serve as a communication relay. A vehicle with a moderately degraded motor/sensor has a capability value of 1 and is deemed unable of performing tracking or surveillance, but can be allocated as a communication relay. Lastly, a capability value of 2 is given any vehicle with severe motor/sensor failures such that it cannot perform any task and must return immediately to base for repairs. The approximate control policy is then computed accounting for this assessment of vehicle capability.

### IV. Flight Results

#### A. Experimental Setup

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 [11]. 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 3, the primary components of the VSTL are: 1) A camera-based motion capture system for reference positions, velocities, attitudes and attitude rates; 2) A cluster of off-board computers for processing the reference data and calculating control inputs; 3) 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 [11].

#### B. Mission Scenario

The mission scenario implemented is a persistent surveillance mission with static \textit{search} and dynamic \textit{track} elements. We employ a heterogeneous team of six agents who begin at some base location \( Y_b \) and are tasked to persistently search the surveillance area. As threats are discovered via search, additional agents are called out to provide persistent tracking of the dynamic threat. Figure 4 shows the initial layout of the mission. Agents 1 – 6 are shown in the base location \( Y_b \), while threats are located in the surveillance region, \( Y_s \).

#### C. Results

In this section, we present the results of flight tests of the multi-agent persistent mission planner in both the Boeing vehicle swarm rapid prototyping testbed [11] and the
Fig. 5. Persistent search and track flight results with six agents and two dynamic threats. Agents rotate in a fuel-optimal manner, (given stochastic models) using a BRE generated policy to keep a sufficient number of agents in the surveillance area to (i) track any known threats and (ii) continually search the surveillance area. Vertical dash-dot lines indicate threat detection events.

MIT RAVEN facility [19]. Flight tests are divided into two scenarios: With and without induced failures.

Figure 5 presents the results for the no-failure scenario, where a persistent surveillance mission is initiated with known agent capabilities and two unknown, dynamic threats. Figure 5 shows the flight location of each of the agents as well as the persistent coverage of the surveillance and communication regions. In this case, two of the threats were detected early, around 20 & 50 time steps into the mission by $UAV_1$. These detection events cause $UGV_1$ and $UAV_2$ to be called out to the surveillance region. At around 60 steps, $UAV_3$ launches from base to relay communications and ultimately take the place of $UAV_1$ before it runs out of fuel. Similarly, $UAV_4$ is proactively launched at roughly 110 steps to replace $UAV_2$. In addition to aerial agents, the ground vehicles also swap in a fuel-optimal manner given the stochastic fuel model detailed in Section II-A.3. It is important to note that “track” tasks are generated by the
discovery of threats whereas the search task is continuous. It can be seen in Figure 5 that the surveillance region is persistently surveyed over the duration of the mission. Also of note in Figure 5, the annotated vertical black dash-dot lines indicate important events during the flight test, such as threat discovery.

Figure 6 presents the results of a second scenario where a similar mission is initiated with initially known agent capabilities and unknown, dynamic threats. In this case however, sensor degradations are induced at the agent-level such that the onboard health monitoring module detects (reactive) a decrease in performance and begins analyzing the extent of the “failure”. A new measure for the capability of the agent is then estimated and sent to the planner so it can be factored into future plans (proactive). In each instance of sensor degradation, the planner receives this capability measure and initiates plans to swap the degraded vehicle with one that is fully-functional. Again, the annotated vertical black dash-dot lines indicate the occurrence of important events during the mission, such as sensor failure and recovery events.

Figure 7 compares the cost function $g$ associated with the MDP planner, as formulated in Section II-A.4, throughout the test flights. The test missions including sensor degradations were averaged to account for inherent stochasticity in the induced failure times. As expected, introducing such events causes an increase in cost, but only slightly over the nominal, no-failure case. This slight increase is due to additional fuel spent swapping the incapable vehicles out for capable ones as well as the additional cost incurred by the degraded sensor in the surveillance region.

V. CONCLUSIONS

This paper has presented an extended formulation of the persistent surveillance problem, which incorporates communication constraints, a stochastic sensor failure model, stochastic fuel flow dynamics, agent capability constraints and the basic constraints of providing surveillance coverage using a team of unmanned vehicles. A parallel implementation of the BRE approximate dynamic programming algorithm was used to approximate a policy. An analysis of this policy via actual flight results indicates that it correctly coordinates the actions of the team of UAVs to simultaneously provide surveillance coverage and communications over the course of the mission, even in the event of sensor failures and reduced agent capabilities.

ACKNOWLEDGMENTS

Research supported by Boeing Research & Technology, Seattle; AFOSR FA9550-08-1-0086; and the Hertz Foundation.

REFERENCES