Health Aware Stochastic Planning For Persistent Package Delivery Missions using Quadrotors

Ali-akbar Agha-mohammadi[1], N. Kemal Ure[1], Jonathan P. How[1] and John Vian[2]*†
‡

August 24, 2014

Abstract

In persistent missions, taking system's health and capability degradation into account is an essential factor to predict and avoid failures. The state space in health-aware planning problems is often a mixture of continuous vehicle-level and discrete mission-level states. This in particular poses a challenge when the mission domain is partially observable and restricts the use of computationally expensive forward search methods. This paper presents a method that exploits a structure that exists in many health-aware planning problems and perform a two-layer planning scheme. The lower layer exploits the local linearization and Gaussian distribution assumption over vehicle level states while the higher level maintains a non-Gaussian distribution over discrete mission-level variables. This two-layer planning scheme allows us to limit the expensive online forward search to the mission-level states, and thus predict system's behavior over longer horizons in the future. We demonstrate the performance of the method on a long duration package delivery mission using a quadrotor in a partially-observable domain in the presence of constraints and health/capability degradation.

1 Introduction

Autonomous robotic systems are being utilized in a wide range of interesting applications by researchers in both the controls and artificial intelligence communities. Such applications include the use of Unmanned Aerial Vehicles (UAVs) for surveillance missions [1], autonomous air-to-ground delivery [2], counter wildlife poaching [3], oil and gas spill detection [4], and firefighting [5]. A common theme among these robotic planning missions is the uncertainty that stems from both the environment and the vehicle

**This work was supported by Boeing Research and Technology

†[1]Ali-akbar Agha-mohammadi, N. Kemal Ure and Jonathan P. How are with Faculty of Aeronautics and Astronautics, Massachusetts Institute of Technology, 77 Mass Ave. Cambridge, MA USA aliagha,ure,jhow@mit.edu

‡[2]John Vian is with Boeing Research and Technology, Seattle WA john.bian@boeing.com

capabilities. This includes the external uncertainties, such as the environmental effects (wind) and variations in the demand/tasks that must be performed, and the internal uncertainties, such as fuel/battery consumption dynamics and actuator/sensor failure dynamics, that impact the vehicle capabilities throughout the mission, which has been described as the *health-aware planning* (HAP) problem [6].

Markov Decision Processes (MDPs) [7] serve as a standard framework to study such stochastic, multi-stage decision making problems. However, MDP formulation assumes that the perfect state information is available to the planner, which is not a realistic assumption for many real-life scenarios, where measurement signals are contaminated with noise. Partially Observable Markov Decision Processes (POMDPs) [8, 9] extend the MDP framework to scenarios where the perfect state information is not available. While [6, 10] developed methods to address this health aware planning problem, they are based on MDPs. However, the POMDP framework is a better fit for this HAP problem because the capability and health states are typically only partially observable. The challenge with any POMDP formulation is developing a computationally tractable solution algorithm. Moreover, the state space associated with the HAP problem is high dimensional due to augmentation of the vehicle's dynamical states with the health states, which renders the development of computationally feasible algorithms very challenging.

The main contribution of this work is to devise a planner for long-endurance missions that require health management in partially-observable domains. This problem requires planning in the belief space (space of distributions) over the vehicle-level states (such as vehicle locations) as well as mission-level states (such as health, capabilities, etc.). The high dimension of this joint space restricts the use of existing forward search methods in the belief space. This paper exploits a decomposition between unstabilizable (i.e. health variables) and stabilizable belief states (i.e. vehicle dynamics) and restricts the online forward search only to unstabilizable belief states to reduce computational complexity. We present the algorithm development for the proposed framework and simulation results for persistent payload delivery missions. The developed algorithm enables motion planning for package delivery missions with persistent duration, limited fuel and stochastic health dynamics. In addition, although the algorithm is validated only for package delivery missions, the general idea can be applied to any planning problem that admits such decomposition, which extends the scale of solvable planning problems that can be formulated as a POMDP.

1.1 Related Work and Method Contributions

Development of optimal [11, 12] and approximate [13, 14] planning algorithms for MDPs have been an active area of research in the artificial intelligence community. However, many real world robotics applications inherently possess imperfect/partial state information and noisy sensory measurements [15], leading to interest in planning with POMDPs [9, 16, 17]. Sampling-based methods are typically used in partially-observable problems to handle constraints and continuous planning domains, and [18–21] are among the pioneering methods in combining sampling-based frameworks and POMDP solvers. However, these methods are valid for a single initial belief and cannot handle real-time replanning from new beliefs. Moreover, some of the heuristics used in these methods do not apply to the HAP problem, for example in HAP we need the

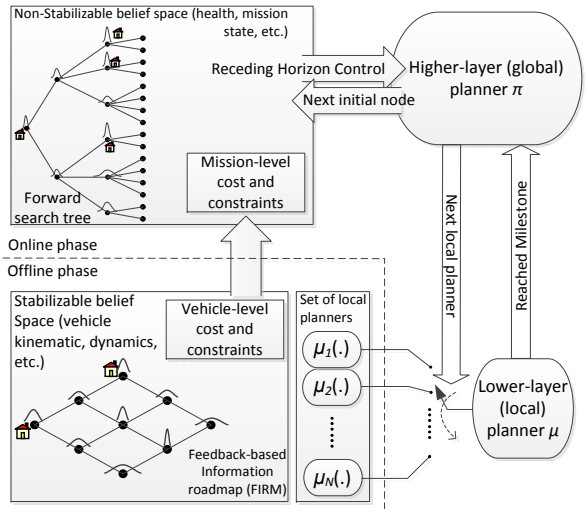

Figure 1: The architecture of the proposed framework for health-aware planning in partially-observable domains.

system to go back to the base or recharge stations periodically and thus we loops cannot be ignored as opposed to the approach in [18]. Recently, the feedback-based information roadmap (FIRM) framework [22, 23], which combines the graph-based methods and belief space planning, enables scalable and online planning in the belief space. However, the framework is only applicable to systems for which the belief state can be stabilized by a feedback controller, which is not the case here because mission-level states such as battery/fuel level and sensor/actuator health status are not directly affected by the lower-level control signals. Existence of such unstabilizable belief states restricts the usage of the FIRM framework in health-aware planning problems. FIRM needs to be extended to be more broadly applicable to handle high dimensional motion planning problems with unstabilizable belief states, such as the HAP problem.

On the other hand, the problem of accounting for sensing hardware health and complete loss-of-vehicle scenarios has been investigated by researchers. Marier et al. [6] has investigated this problem in the context of health-aware coverage control with small UAVs. Nigam et al. [1] have developed a control architecture for achieving persistent surveillance with multiple UAVs under limited fuel and stochastic health dynamics. However the proposed control improvement method is only *reactive* to the sensor and actuator failures. In the perfect state information case, Ure et. al [10] have developed a health-aware planning framework, which combines the trajectory sampling approximate MDP solvers [24] and learning based adaptive control [25]. This work aims to extend the health-aware planning methods to partially-observable domains to handle challenging tasks such as persistent package delivery in a partially-observable setting with health dynamics. The diagram of the proposed framework is displayed in Fig. 1.

The main idea behind the proposed framework is to reduce the computational cost of the online forward search (online prediction of system's performance) via reducing the dimension of the belief subset for which one needs to perform an online forward

search. The proposed framework exploits an existing structure in health-aware planning missions and proposes a two-layer planning framework, by decomposing the state space into two parts: the vehicle-level and mission-level parts. The vehicle-level state dynamics are locally well approximated by linear systems with Gaussian noises, while the mission-level states typically live in discrete spaces with non-Gaussian distributions. In health-aware planning, vehicle-level dynamics usually consists of position, orientation, as well as translation and angular velocities of the vehicle, and mission-level variables correspond to the vehicle's fuel/battery level, health and capability indicators, etc.

The lower layer of the planner utilizes the information roadmap algorithm, which has been shown to be highly scalable through both theoretical and simulation results [23]. Higher-level planner, incorporates the fuel and health dynamics into the planning by performing a forward search in the space of mission-level variables. This forward search is performed in a receding horizon control (RHC) scheme, where both vehicle-level and mission-level costs and constraints are taken into account. However, during the repeated forward search in RHC, the costs and constraints associated with the vehicle-level variables do not need to be re-evaluated at every step; those costs and constraints are computed offline via an information roadmap and then retrieved by the higher-level planner as needed. To the best of our knowledge, this is the first POMDP-based planner that addresses the health-awareness in persistent planning problems.

2 Problem Definition

In this section, we describe the elements of the problem formulation.

2.1 Preliminaries

Let us denote the system state, control, and observation signals at the k-th time step respectively by $x_k \in \mathbb{X} \subseteq \mathbb{R}^{d_x}$, $u_k \in \mathbb{U} \subseteq \mathbb{R}^{d_u}$, and $z_k \in \mathbb{Z} \subseteq \mathbb{R}^{d_z}$, where $\mathbb{R}^n$ denotes the n-dimensional Euclidean space. $\mathbb{X}$, $\mathbb{U}$, and $\mathbb{Z}$ denote the state, control, and observation spaces, respectively.

Let us describe the state evolution model by the transition probability $\mathbb{P}(X|x,u) := \Pr(x_{k+1} \in X | x_k = x, u_k = u)$ where $X \subset \mathbb{X}$, $x \in \mathbb{X}$, and $u \in \mathbb{U}$. Also, in the presence of noisy measurements, we describe the observation model by the likelihood model $\mathbb{P}(Z|x,u) := \Pr(z_k \in Z | x_k = x, u_k = u)$, where $Z \subset \mathbb{Z}$. In the presence of noise in measurement signal, an estimation module can provide a distribution over all possible states, referred to as belief, for decision making purposes. Formally, belief at the k-th time step is defined as $b_k = \mathbb{P}(X|z_{0:k}, u_{0:k-1}) := \Pr(x_k \in X | z_{0:k}, u_{0:k-1})$. We denote the belief space by $\mathbb{B}$. The Bayesian estimator evolves the belief recursively and one can denote the belief evolution by its transition probability $\mathbb{P}(B|b,u) := \Pr(b_{k+1} \in B | b_k = b, u_k = u)$, where $B \subset \mathbb{B}$ and $b \in \mathbb{B}$.

State Decomposition: In this work, we consider a class of systems, where the state can possibly be a mixture of continuous and discrete variables. More rigorously speaking, we consider systems where state consists of two parts $x_k = (x_k^s, x_k^n)$, referred to as stabilizable and non-stabilizable parts, respectively.

• *Stabilizable part (vehicle-level variables)*: x^s is the part of the state vector, for which we can design feedback controllers to steer the pdf over x^s (denoted by b^s) to predefined beliefs [23]. As we discuss in experiments section, usually stabilizable states correspond to continuous variables (e.g. vehicles location, orientation, etc.) which has a well-behaved (e.g. well-linearizable) evolution model in the sense that one can design appropriate controllers to regulate b^s.

• *Non-Stabilizabile part (mission-level variables)*: x^n is the part of state vector that is not stabilizable. As we discuss in experiments section, x^n often corresponds to the discrete part of the space for which designing belief regulating controllers is challenging. Furthermore, x^n consists of parts of the state describing the environment (e.g., a target state) over which we have no control. In health-aware planning x^n may represent higher level properties such as health of the system, remaining fuel, state of a task, or vehicle's capability to perform certain functions or tasks. The evolution of non-stabilizable part may depend on the stabilizable part (e.g., health or capability may depend on the location of the vehicle and applied controls). To perform a proactive planning one needs to maintain a distribution over these states and incorporate them into the planning scheme.

Vehicle-level transitions: In this work it is assumed that the local evolution of the vehicle-level variables x^s is independent of mission-level variables. In other words, one can consider the control space $\mathbb{U}^s$ and observation space $\mathbb{Z}^s$ (as lower-dimensional projections of $\mathbb{U}$ and $\mathbb{Z}$) which induce a transition probability $\mathbb{P}(X^s|x^s, u^s)$, where $X^s \subset \mathbb{X}^s$, $x^s \in \mathbb{X}^s$, and $u^s \in \mathbb{U}^s$. Accordingly, the distribution over x^s is denoted by $b^s_k \in \mathbb{B}^s$, which is evolved by transition probability $\mathbb{P}(B^s|b^s, u^s)$, where $B^s \subset \mathbb{B}^s$ and $b^s \in \mathbb{B}^s$. As a result, to generate u^s one can design a feedback law μ^s that is a mapping from $\mathbb{B}^s$ to $\mathbb{U}^s$.

Mission-level transitions: According to above definitions, one can decompose the control space as $\mathbb{U} = \mathbb{U}^s \times \mathbb{U}^n$, the observation space as $\mathbb{Z} = \mathbb{Z}^s \times \mathbb{Z}^n$. However, it is important to note that such a separation does *not* induce separated transitions for mission-level state. In other words, mission-level state dynamics depend on the *full* state, control, and observation vectors. For example state transition is described as $\mathbb{P}(X^n|x, u)$, where $X^n \subset \mathbb{X}^n$, $x \in \mathbb{X}$, and $u \in \mathbb{U}$. Therefore, for a set $X = X^s \times X^n$, according to the belief definition and conditional probability rules, we can write

$$
\begin{aligned}
b_k &= \Pr(x_k \in X | z_{0:k}, u_{0:k-1}) \\
&= \Pr(x^n_k \in X^n | x^s_k \in X^s, z_{0:k}, u_{0:k-1}) \Pr(x^s_k \in X^s | z_{0:k}, u_{0:k-1}) \\
&= \underbrace{\Pr(x^n_k \in X^n | x^s_k \in X^s, z_{0:k}, u_{0:k-1})}_{b^n_k} \underbrace{\Pr(x^s_k \in X^s | z^s_{0:k}, u^s_{0:k-1})}_{b^s_k}
\end{aligned}
\tag{1}
$$

Accordingly, the distribution over x^n, denoted by $b^n_k \in \mathbb{B}^n$, is evolved as $\mathbb{P}(B^n|b, u)$, where $B^n \subset \mathbb{B}^n$ and $b \in \mathbb{B}$. As a result, a feedback law μ^n to generate u^n has to map the *full* belief space to u^n, i.e., $u^n = \mu^n(b)$. In the present setting, we pick the weighted Dirac mixture approximation to represent the non-Gaussian distribution $b^n = \mathcal{D}(\{x^n(m)\}_{m=1}^M, \{w(m)\}_{m=1}^M)$. The corresponding pdf can explicitly be written as $\sum_{m=1}^M w(m)\delta(x^n - x^n(m))$, where $\delta(\cdot)$ denotes the Kronecker delta function and $w(m)$ is the weight of the m-th particle that satisfies the normalization constraint $\sum_{m=1}^M w(m) = 1$.

2.2 Problem statement

In this paper, we consider the problem of health-aware planning, where we define our problem in two parts: proactive planning and reactive planning.

2.2.1 Proactive Planning

In the proactive part, we predict the evolution of system's state, health, and capabilities in future time steps. Accordingly, a plan is generated to accomplish a given task while respecting constraints on system health and capability. The cost associated with the stabilizable part is reflected in $c^s(b^s, u^s)$ and the cost over health/capability variables is reflected in $c^n(b^n, u)$. The overall cost is a combination of these two costs $c(b, u) = g(c^s(b^s, u), c^n(b^n, u))$ where g is assumed to be a linear map.

Constraints on the vehicle-level and mission-level states are formalized by adding appropriate chance constraints to the optimization problem. $F^s \subset \mathbb{X}^s$ and $F^n \subset \mathbb{X}^n$ denote the undesirable set of states and ϵ^s and ϵ^n denote the maximum chance of hitting these states. Accordingly, the full planning problem is stated as follows:

$$
\begin{aligned}
&\min_{\Pi} \mathbb{E} \sum_{k=1}^{\infty} \gamma^k c(b_k, \pi(b_k)) \\
&s.t. \quad b_{k+1} \sim \mathbb{P}(B_{k+1}|b_k, u_k), \quad x^s \notin S \subset \mathbb{X}^s \\
&\qquad b^n_{k+1} = \tau^{station}(b^n_k), \quad x^s \in S \subset \mathbb{X}^s \\
&\qquad \Pr(x^s_k \in F^s | z_{0:k}, u_{0:k-1}) < \epsilon^s, \ \ \forall k \\
&\qquad \Pr(x^n_k \in F^n | z_{0:k}, u_{0:k-1}) < \epsilon^n, \ \ \forall k
\end{aligned} \tag{2}
$$

where S denotes the location of the stations such as "base", "recharge", or "delivery" stations in persistent package delivery mission. In these stations, system's belief is governed by a deterministic function $\tau^{station}$: At the base station, health is reset to its maximum value. Similarly, the vehicle gets refuelled in recharge stations. The packages are picked-up in the base station and delivered at the delivery station.

2.2.2 Reactive Planning

In the reactive phase, we require to solve the proactive problem online. In other words, if system encounters any unexpected large deviations, health degradation, or failures, we need to be able to recover in real-time and resolve the proactive POMDP problem in (2). It is well known that such a reactive behavior in partially-observable environments is very challenging due to the curse of history [26]. In other words, replanning in partially-observable spaces requires all the computation to be reproduce.

In this research, by splitting latent variables to stabilizable and non-stabilizable parts, we show that we only need to predict the behavior of the non-stabilizable variables online. Therefore, for systems with a small dimension of non-stabilizable variables, the reactive planning becomes feasible.

3 The Algorithm

To accomplish a given mission in the presence of uncertain mission dynamics and to provide a computationally tractable version of the optimization in (2), we propose a hybrid planning framework. We sample a set of intermediate milestones, and perform a two-level planning over this set of milestones as described further below.

• *Global planner*: The global decision making algorithm provides a feedback (or open-loop) plan over the milestones according to both *(i)* high-level states such as vehicle's health, capability, and the mission state, and *(ii)* the costs that come from low-level planning such as available information for control the vehicle and the distribution over the error in following a reference trajectory. Hence, at every milestone, taking into account both mission-level and vehicle-level variables, the global planner, denoted by π, decides which local controller has to be utilized at the current milestone. The selected local controller in turn guides the system toward the next milestone (see Fig. 1). Therefore, denoting the set of milestones by $\mathcal{M} = \{\mathbf{m}^i\}$ and the set of local-planners by $\mathcal{L} = \{\mu^j(\cdot)\}$, the higher level planner can be written as a mapping $\pi : \mathcal{M} \to \mathcal{L}$.

• *Local planner*: Once the local planner $\mu(\cdot)$ is fixed, it drives the system to the next milestone associated with μ. Each local controller $\mu(\cdot)$ consists of two parts $\mu = (\mu^s, \mu^n)$ where μ^s and μ^n generate the control signals for the lower-level and higher-level variables, respectively, as discussed in Section 2.1. While the local controller generates control signals, the probability distribution over the higher-level variables is propagated under the action of μ. Accordingly, when the system *(i)* reaches the next milestone or *(ii)* the health- or mission-monitoring module reports a significant deviation from the nominal plan, the system control is handed over to the higher-level planner π.

3.1 Local Planners

Decomposing the state space to stabilizable and non-stabilizable parts $\mathbb{X} = \mathbb{X}^s \times \mathbb{X}^n$, one can generate an information roadmap (IRM) [23] in $\mathbb{X}^s$. An information roadmap is a representative graph in the belief space whose nodes are certain probability distributions and whose edges are feedback controllers that takes the belief from one node to another.

To generate an IRM in $\mathbb{X}$, we first sample a set of N points (milestones) $\{\mathbf{m}^i\}_{i=1}^N$ from the constraint-free space $\mathbb{X}^s \setminus F^s$. Utilizing an appropriate metric, we connect each $\mathbf{m}^j$ to its k-nearest neighbors in the constraint-free space $\mathbb{X}^s \setminus F^s$, to generate a representative graph. Figure 2 shows a simple graph with nine nodes.

Sampling belief nodes: To leverage milestones in the state space to milestones in belief space, we first linearize the system about each $\mathbf{m}^j$. Then, for each $\mathbf{m}^j$ we design an SLQG (Stationary Linear Quadratic Gaussian) controller as the local controller μ^j. Accordingly, one can characterize a unique normal distribution $b^{s,j} = \mathcal{N}(\mathbf{m}^j, P^j)$ which is reachable under μ^j [23]. Covariance matrix P^j is the solution of the Riccati equation at $\mathbf{m}^j$ that can be solved efficiently. Figure 2 shows a cartoon of set of sampled beliefs. We denote the set of samples in the state space by $\mathcal{M} = \{\mathbf{m}^j\}$ and the set of samples in the belief space by $\mathbb{V} = \{b^{s,j}\}$.

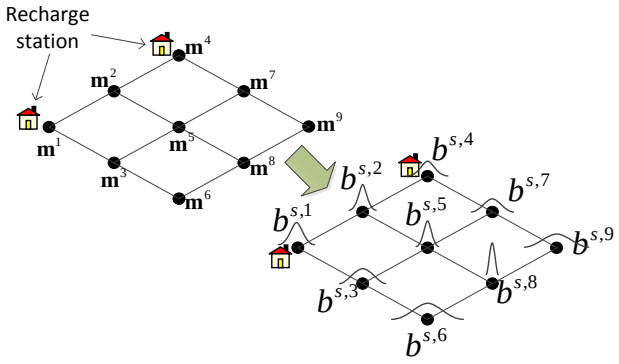

Figure 2: This figure in left shows a set of sampled milestones in $\mathbb{X}^s$. The figure in right shows the unique distributions associate with each milestone.

Connecting belief nodes: Since these belief nodes are computed offline, we can compute the edge costs in the belief space offline as well. Therefore, for an edge going from $b^{s,i}$ to $b^{s,j}$, we simulate the system evolution under the local controller μ^j and compute the following edge cost:

$$c^s(b^{s,i}, \mu^j) := \mathbb{E} \sum_{k=0}^{\mathcal{T}} c(b_k^s, \mu^j(b_k^s)), \quad b_0^s = b^{s,i} \tag{3}$$

where $\mathcal{T}$ is the time where $b_{\mathcal{T}}^s$ enters into an ϵ-ball around $b^{s,j}$. The expectation in (3) can be computed through Monte Carlo simulations as all these computations are carried out offline. We denote the set of all edges (local planners) by $\mathcal{L} = \{\mu^j\}$ and the set of edges outgoing from the i-th node by $\mathcal{L}(i)$.

3.2 Global planner

In this section, we design the global planner that incorporates the notion of the health/capability into planning. In general, it is desirable to have such a graph in the entire belief space, solve a Dynamic Programming on the graph, and compute a closed-loop feedback solution $\pi : \mathbb{V} \to \mathbb{M}$ offline [23]. However, due to the challenging task of designing globally reachable beliefs for variables such as health and capability, constructing a graph in non-stabilizable belief space is a formidable task. Consequently, we rely on a forward search to generate a search tree and find the best sequence of edges $\mu_{0:\infty}$ that minimizes the cost function and respects the constraints.

Search tree: Each node $\mathbf{n}$ of the search tree is a probability distributions b over systems state which is stored in two parts $\mathbf{n}[b^s]$ and $\mathbf{n}[b^n]$ based on Eq. (1). At any given node $\mathbf{n}[b^s] \in \mathbb{V}$ and $\mathbf{n}[b^n] \in \mathbb{B}^n$. To construct the tree, we expand the root node by selecting local controllers from set of edges outgoing from $\mathbf{n}[b^s]$ in the underlying FIRM, and propagating the probability distributions to get the child nodes in the tree, and repeat the same procedure from the child nodes to construct the tree. Figure 3 shows such a search tree over non-stabilizable belief space for the search depth of

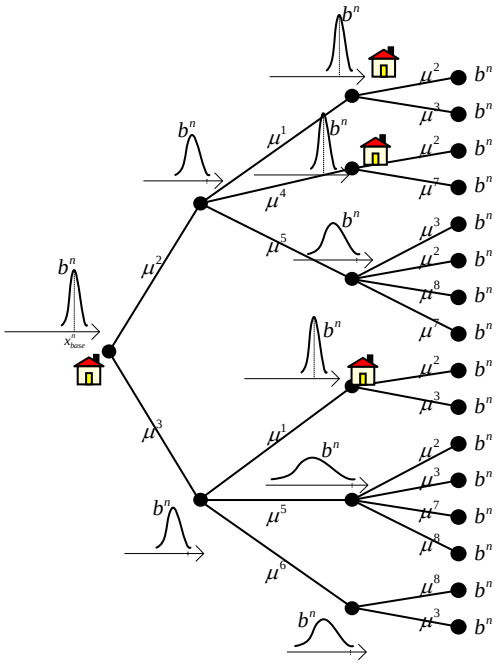

Figure 3: Illustrates the search tree in the non-stabilizable belief space for the depth of three (corresponding to the graph in Fig. 2). Taking edges μ^1 and μ^4 drives the system to base station, where the non-stabilizable belief gets reset.
three.

Belief propagation along tree edges: Taking edge (or local controller) μ^j at node $\mathbf{n}_k$ we propagate belief $\mathbf{n}_k[b^n]$ using a particle filter over the edge to get the belief trajectory $b^n_{0:\mathcal{T}}$, where $b^n_0 = \mathbf{n}_k[b^n]$. Belief $b^n_{\mathcal{T}}$ at the end of edge will define the next node, i.e., $\mathbf{n}_{k+1}[b^n] = b^n_{\mathcal{T}}$. In generating $b^n_{0:\mathcal{T}}$, we rely on maximum likelihood assumption for the unknown variables in future time steps as a typical assumption in belief space planning [18, 27]. It is important to note that there is no need to propagate b^s as it is known a priori that $\mathbf{n}_{k+1}[b^s] = b^{s,j}$ due to the FIRM structure. Also, since this propagation is done offline, we consider all possible values for future measurements and there is no maximum likelihood assumption in propagating b^s. We denote the node propagation (or distribution propagation) along edge μ by $\mathbf{n}_k \xrightarrow{\mu} \mathbf{n}_{k+1}$.

Edge cost: To search over the tree, we associate a cost to each transition on the tree. The cost associated with taking edge μ at node $\mathbf{n}$ (with joint belief $b_0 = \mathbf{n}[b^n_0]\mathbf{n}[b^s_0]$) is the total cost that incurs under the local planner $\mu = (\mu^s, \mu^n)$:

$$
\begin{aligned}
c(\mathbf{n}, \mu) &= \mathbb{E}\sum_{t=0}^{\mathcal{T}} c(b_t, u_t) = g(\mathbb{E}\sum_{t=0}^{\mathcal{T}} c^s(b^s_t, u^s_t), \mathbb{E}\sum_{t=0}^{\mathcal{T}} c^n(b^n_t, u_t)) \\
&= g(c^s(b^s_0, \mu^s), c^n(b^n_0, \mu)), \quad b^s_0 = \mathbf{n}[b^s],\ b^n_0 = \mathbf{n}[b^n] \qquad (4)
\end{aligned}
$$

where $c^s(b^s_0, \mu^s)$ is the cost of taking local controller μ^s at belief b^s_0. Similarly, $c^n(b^n_0, \mu)$

denotes the cost of taking local controller μ^n at belief b_0^n. c^s is computed offline as discussed in (3), but c^n is computed online.

RHC-based formulation: At every step, receding horizon control (RHC) algorithm performs a two-stage procedure. At the first stage, RHC computes an open-loop sequence of edges over a finite horizon H. In the second stage, it only executes the first edge in the sequence and discards the rest. This two-stage procedure is repeated from the new node. The first stage, i.e., finding the optimal sequence of edges $\mu_{0:H}$ from a given node $\mathbf{n}$ is formulated as minimizing the following cost-to-go function, whose cost function and constraints are induced by the original problem in (2):

$$\begin{aligned}
&\mu^*_{0:H} = \pi(\mathbf{n}) = \arg\min_{\mu_{0:H}} \mathbb{E} \sum_{k=1}^{H} c(\mathbf{n}_k, \mu_k), \;\; \mathbf{n}_0 = \mathbf{n} \\
&s.t.: \quad \mathbf{n}_k \overset{\mu_k}{\rightarrow} \mathbf{n}_{k+1}, \;\; \forall k \\
&\Pr(x_t^s \in F^s | z_{0:t}, u_{0:t-1}) < \delta^s, \; x_0^s = \mathbf{n}_k[x^s], \; t \leq \mathcal{T}(\mu_k) \\
&\Pr(x_t^n \in F^n | z_{0:t}, u_{0:t-1}) < \delta^n, \; x_0^n = \mathbf{n}_k[x^n], \; t \leq \mathcal{T}(\mu_k) \\
&\mathbf{n}_k[b^n] \leftarrow \tau^{station}(\mathbf{n}_k[b^n]), \;\; \text{if } \; \mathbf{n}_k[b^s] \in \mathbb{V}^{station}
\end{aligned} \tag{5}$$

where $c(\mathbf{n}_k, \mu_k)$ is the cost associated with taking the edge μ_k at node $\mathbf{n}_k$. The node $\mathbf{n}_{k+1}$ is the resulting node of taking edge μ_k at node $\mathbf{n}_k$. $\mathbb{V}^{station} \subset \mathbb{V}$ is the set of FIRM nodes that correspond to milestones lying in station areas S.

Efficient search via decomposition: To find $\mu^*_{0:H}$ we need to generate a forward search tree of depth H at every step in the space of probability distributions. However since the tree grows exponentially (see Fig. 3), computing the costs $c(\mathbf{n}_k, \mu_k)$ and evaluating constraints in (5) is computationally very expensive, in particular for high-dimensional systems. However, by decomposing the state space to stabilizable and non-stabilizable parts, we limit the forward search to the non-stabilizable part. Therefore, in problems such as the health-aware planning problem, where the non-stabilizable part has a much lower dimension compared to the full belief space, the forward search can be performed efficiently. The cost decomposition is carried out based on (4), where c^s and c^n denote the cost associated with the lower-level and higher-level variables, respectively. All c^s parts can be retrieved from the stored values and do not need to be recomputed. Similarly, the probability of violating lower level constraints F^s can be computed offline, while the probability of violating higher level constraints F^n needs to be computed online.

Cost-to-reach: To perform the forward search, we assign a cost-to-reach to each node of the search tree. The cost-to-reach associated with node $\mathbf{n}$ on the tree is denoted by $\mathbf{n}[ctr]$, which indicates the cost of reaching node $\mathbf{n}$ starting from the root.

$$\mathbf{n}_k[ctr] = \mathbf{n}^p_{k-1}[ctr] + c(\mathbf{n}^p_{k-1}, \mu^{\mathbf{n}^p, \mathbf{n}}) \tag{6}$$

where $\mathbf{n}^p$ is the parent node of $\mathbf{n}$ in the tree, and $\mu^{\mathbf{n}^p, \mathbf{n}}$ is the edge connecting $\mathbf{n}^p$ to $\mathbf{n}$.

Search algorithm: To solve the RHC optimization in (5), we perform a forward search in the space non-stabilizable beliefs. However, we generate the search tree by performing the search using the macro-actions introduced by the underlying graph in the state space. Algorithm 1 illustrates the procedure.

Algorithm 1: Forward search to solve RHC optimization

input : Initial stabilizable belief b_0^s, Initial non-stabilizable belief b_0^n, Search depth H
output : Next edge (or local controller) μ
Set $\mathbf{n}_0[b^s] = b_0^s,\ \mathbf{n}_0[b^n] = b_0^n,\ \mathbf{n}_0[ctr] = 0,\ \mathbf{n}_0[P^s] = 0,\ \mathbf{n}_0[P^n] = 0,\ \mathbf{n}_0[path] = \emptyset$;
Set OPEN $= \{\mathbf{n}_0\}$;
while $\min(|\mathbf{n}[path]|) < H$ **do**
    Pop $\mathbf{n}$ from OPEN, where $\mathbf{n} = \arg\min_{\bar{\mathbf{n}} \in \text{OPEN}} |\bar{\mathbf{n}}[path]|$;
    foreach μ outgoing from $\mathbf{n}$ **do**
        Propagate belief b^n under μ to get the belief trajectory $b^n_{0:\mathcal{T}}$;
        Set $\mathbf{n}'[b^n] = b^n_{\mathcal{T}}$ and set $\mathbf{n}'[b^s]$ to the target node of μ in FIRM;
        Retrieve $P^s = \Pr(x^s_{0:k} \in F^s)$ from FIRM;
        Compute $P^n = \Pr(x^n_{0:k} \in F^n)$;
        Compute $\mathbf{n}'[P^s] \leftarrow 1 - (1 - \mathbf{n}[P^s])(1 - P^s)$;
        Compute $\mathbf{n}'[P^n] \leftarrow 1 - (1 - \mathbf{n}[P^n])(1 - P^n)$;
        if $\mathbf{n}'[P^s] < \epsilon^s$ ***and*** $\mathbf{n}'[P^n] < \epsilon^n$ **then**
            Retrieve $c^s(\mathbf{n}[b^s], \mu^s)$ from FIRM;
            Compute $c^n(\mathbf{n}[b^n], \mu)$ based on $b^n_{0:\mathcal{T}}$;
            $\mathbf{n}'[ctr] = \mathbf{n}[ctr] + g(c^s, c^n)$;
            $\mathbf{n}'[path] = \mathbf{n}[path] \cup \{\mu\}$;
            Push $\mathbf{n}'$ to OPEN;
Find $\mathbf{n}^* \in$ OPEN which has the minimum $\mathbf{n}[ctr]$;
return μ^* as the first edge in $\mathbf{n}^*[path]$;

4 Simulation Results

4.1 Persistent Package Delivery Mission

The objective of the mission is to persistently deliver packages from the base to delivery locations via a quadrotor UAV with limited battery life. The battery voltage decreases while the quadrotor is operational, and the rate of decrease depends on the amplitude of the wind gusts at quadrotor's current location. Whenever the quadrotor lands on a recharge station, the current battery is replaced with a fresh battery, hence the mission can continue indefinitely if the quadrotor reaches to a recharge station before the current battery is dead. In addition, quadrotor's actuators are subject to degradation and the probability of landing successfully on a delivery location is coupled directly with the the actuator's health status. It is assumed that when the quadrotor is at the base, all failures are repaired and the actuators are restored to their original state. Moreover, the battery level and the actuator health status is not directly available and needs to be estimated from noisy measurements. The overall objective of the planner is to compute a policy such that the quadrotor makes trips between the base and delivery locations persistently, while ensuring that quadrotor does not crash because of a depleted battery

and the actuators are healthy enough to perform successful landings. In the subsequent subsections, we provide the mathematical model of the mission.

4.1.1 State and Action Space

The stabilizable states (see Section 2) x^s consists of $(\mathtt{p}_x, \mathtt{p}_y, \mathtt{p}_z, \dot{\mathtt{p}_x}, \dot{\mathtt{p}_y}, \dot{\mathtt{p}_z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi})$, where $\mathtt{p} = (\mathtt{p}_x, \mathtt{p}_y, \mathtt{p}_z) \in \mathbb{R}^3$ denotes the position of the quadrotor and $(\phi, \theta, \psi) \in [-\pi, \pi]^3$ denotes the orientation of the quadrotor. The input space of the quadrotor consists of $u = (\delta_{collective}, \delta_{roll}, \delta_{pitch}, \delta_{yaw})$, where inputs are the collective, roll, pitch and, yaw commands, respectively. The non-stabilizable states x^n consists of $x^n = (v, \zeta, d)$, where $v \in [0, v_{max}]$ is the voltage-drop across the battery with v_max being the maximum voltage. $\zeta \in \{0, 1, 2, ..., \zeta_{max}\}$ is the actuator health status where ζ_{max} is the maximum actuator health and $d \in \{0, 1\}$ is a binary variable that indicates if the quadrotor is carrying a package or not.

4.1.2 State Transition Model

For a fixed high-level variables, the continuous stochastic state transition model of the lower-level states x^s is given as:

$$\dot{x^s} = f^s(x^s, u^s) + w, \quad w \sim \mathcal{N}(0, Q), \tag{7}$$

where w is the vehicle motion noise drawn form a zero-mean Gaussian distribution with covariance matrix Q. The function f^s is given as [28].

$$\begin{aligned} f^s(x^s, u) = (\dot{\mathtt{p}_x}, \dot{\mathtt{p}_y}, \dot{\mathtt{p}_z}, g\phi, -g\theta, m^{-1}\hat{\delta}_{collective}, \dot{\phi}, \dot{\theta}, \dot{\psi}, \\ LI_x^{-1}\delta_{roll}, LI_y^{-1}\delta_{pitch}, I_z^{-1}\delta_{yaw})^T, \end{aligned} \tag{8}$$

where $g = 9.81 m/s^2$ is the gravitational constant, $m = 0.650$ (Kg) is the mass of the quadrotor, $L = 0.23$ (meter) is the length of the quadrotor and $(I_x, I_y, I_z) = (0.0075, 0.0075, 0.013)\ Kg \cdot m^2$ are moments of inertia of the quadrotor.

The stochastic state transition model of the non-stabilizable part is given as follows. Let us denote the region corresponding to the recharge station by $X_{rech} \subset \mathbb{X}^s$. Whenever quadrotor is not in a recharge station, i.e., $x^s \notin X_{rech}$, the battery dynamics are given as follows,

$$\dot{q} = \begin{pmatrix} \frac{1}{C_b R_p} & -\frac{1}{C_{cp} R_p} & \frac{1}{C_s R_p} \\ \frac{1}{C_b R_p} & -\frac{1}{C_{cp} R_p R_{cp}} & \frac{1}{C_s R_p} \\ \frac{1}{C_b R_p} & \frac{1}{C_{cp} R_p} & \frac{1}{C_s R_p} \end{pmatrix} q + (i + w^q)\mathbf{1}_{3\times 1}, \tag{9}$$

where, q is the charge stored across the three capacitors in the equivalent circuit model , C_b, C_{cp}, C_s and R_p, R_b, R_{cp} are the capacities and resistances of the equivalent circuit model of the battery [29], i is the current drawn from battery and $w^q \sim \mathcal{N}(0, Q^q)$ is the process noise with intensity Q^q. $\mathbf{1}_{3\times 1}$ is a three-by-one vector of ones. The process noise has a greater intensity in windy areas as the system will draw more current depending on the deviations caused by the wind. This dependency will be discussed further below. The voltage-drop across the battery is given by $v = (C_b^{-1}, C_p^{-1}, C_s^{-1})q$.

If the quadrotor is at a recharge station, i.e., $x^s \in X_{rech}$, the current battery is replaced with a full battery, and v will be reset to its maximum value.

The actuator health dynamics are given as follows. Let us denote the base region by $X_{base} \subset \mathbb{X}^s$ and the actuator health by ζ_k at the k-th time step, where $\{0, 1, 2, ..., \zeta_{max}\}$ is the space of all possible actuator health states. The health transition model is described by:

$$\begin{aligned} &\Pr(\zeta_{k+1} = x | \zeta_k = x, x_k^s \notin X_{base}) = p_a \\ &\Pr(\zeta_{k+1} = x - 1 | \zeta_k = x, x_t^s \notin X_{base}) = 1 - p_a \end{aligned} \tag{10}$$

where $p_a \in [0, 1]$ is the probability that actuator's health will stay at the same level at the next time step. If $\zeta_k = 0$, then actuator is completely failed and the quadrotor is not capable of flying. If the quadrotor is at the base, then the actuator is restored to it's original state, that is $\Pr(\zeta_{k+1} = \zeta_{max} | x_k^s \in X_{base}) = 1$.

Finally, let us represent the locations of the delivery zones by $X_{del} \subset \mathbb{X}^s$. The package dynamics are given as follows,

$$\begin{aligned} &\Pr(d_{k+1} = 1 | d_k = 0, x_k^s \in X_{base}) = \zeta_k \zeta_{max}^{-1} \\ &\Pr(d_{k+1} = 0 | d_k = 1, x_k^s \in X_{del}) = \zeta_k \zeta_{max}^{-1}. \end{aligned} \tag{11}$$

that is the probability of making a successful delivery or pick-up depends on the actuator health of the quadrotor.

4.1.3 Observation Model

In the posed problem of persistent package delivery, the measurements $z = (z^s, z^n)$ consist of the vehicle-level state measurements z^s and mission-level state measurements z^n.

Vehicle-level measurements: Depending on the vehicle's location, the vehicle may have access to absolute state information, or use landmarks to perform delivery in GPS-denied environments. We assume there exist n_l landmarks with known location in the environment that are mainly distributed GPS-denied parts of the environment and in the vicinity of delivery location to ensure a precise package delivery. Therefore, we have $z^s = (z^{abs}, {}^1z, {}^2z, \cdots, {}^{n_l}z)$ where, the absolute observation is $z^{abs} = x^s + v^{abs}$ with Gaussian noise $v^{abs} \sim \mathcal{N}(0, R^{abs})$ with constant covariance R^{abs}. ${}^iz$ the measurement from the i-th landmark consists of relative distance and bearings to the landmark, and its noise intensity is proportional to the distance to i-th landmark. The 3D location of the i-th Landmark is defined as $L^i = [L_x^i, L_y^i, L_z^i]^T$. Denoting the relative vector from robot to landmark L^i by ${}^i\mathbf{d}^g = [{}^id_x^g, {}^id_y^g, {}^id_z^g]^T := L^i - \mathtt{p}$, where $\mathtt{p} = [\mathtt{p}_x, \mathtt{p}_y, \mathtt{p}_z]^T$ is the position of the robot in the ground frame. The relative vector ${}^i\mathbf{d}^g$ needs to be rotated from the ground frame to the body frame by the rotation matrix $R_{bg} = R_{gb}^T$. Thus, ${}^i\mathbf{d}^b = R_{bg}\,{}^i\mathbf{d}^g$.

Mission-level measurements: The measurements associated with x^n are shown by $z^n = (z^v, z^\zeta, z^d)$. It is assumed that the only voltage drop is measurable form the battery dynamics hence $z^v = v + \tilde{v}$, $\tilde{v}$ is the zero mean Gaussian noise with variance R^v. The actuator health ζ is measured by a diagnostic system [30]. The measurement

signal at the k-th time step is denoted by $z_k^\zeta = \zeta_k + \tilde{\zeta}_k$, where the measurement error $\tilde{\zeta}_k$ is modeled as the following nonGaussian distribution $\Pr(\tilde{\zeta}_k = -1) = \Pr(\tilde{\zeta}_k = 1) = p_h$ and $\Pr(\tilde{\zeta}_k = 0) = 1 - 2p_h$. where $0 \leq p_h \leq 0.5$ is the probability that the diagnostic system under- or over-estimate the vehicle health. Finally, it is assumed that d_k is fully observable, i.e., $z_k^d = d_k$. In other words, the vehicle is perfectly aware if it is carrying a package or not at all times.

4.1.4 Cost Function

In the current simulations, the cost function only depends on the mission-level variables. We assume the agent collects 0 cost whenever it delivers or picks-up a package, and it collects 1 cost at every other state.

4.1.5 Constraints

In addition to constraints over the vehicle-level states (e.g. avoiding obstacles), the following chance constraints are put on battery and actuator dynamics: $\Pr(v_k < 0.01) < 0.001,\ \Pr(\zeta_k = 0) < 0.001$. In other words, the probability that battery is completely consumed and the actuator is completely failed has to be small.

4.2 Simulation Results

This subsection provides simulation results for the proposed health-aware planning framework for persistent package delivery missions. Figure 4 shows an urban environment with 9 obstacles. A PRM with 14 nodes and 46 edges, which approximates the connectivity of the constraint-free space, is shown in blue. One of the nodes is associated with base, two nodes with recharge stations, and one node with the delivery location. Packages are initially in the base location and need to be picked up and delivered to the delivery location.

Wind model: The darker regions in Fig. 4 correspond to windy areas. The bell shaped function $\mathcal{W}(x^s)$ retunes the wind intensity for any given location x^s of quadrotor. The standard deviation of the fuel consumption noise is linearly related to the wind intensity $Q^q = (\sqrt{Q^q_{bias}} + \eta\mathcal{W}(x))^2$, where Q^q_{bias} is a part of covariance that is independent of the wind.

An any given position of the vehicle x^s, wind magnifies the intensity of the noise by teh value of the function $\mathcal{W}(\cdot)$ evaluated at x^s. For a one dimensional wind, where the wind center is denoted by c and the wind width is denoted by w, $\mathcal{W}$ is computed via the following bell-shaped function modeling the wind whose decay slope is specified by parameter s:

$$\mathcal{W}(x) = m^{-1}\left(\frac{1}{1+e^{-s(x-c_l)}} - \frac{1}{1+e^{-s(x-c_r)}}\right) \tag{12}$$

where

$$c_l = c - 0.5w,\ \ c_r = c + 0.5w \tag{13}$$

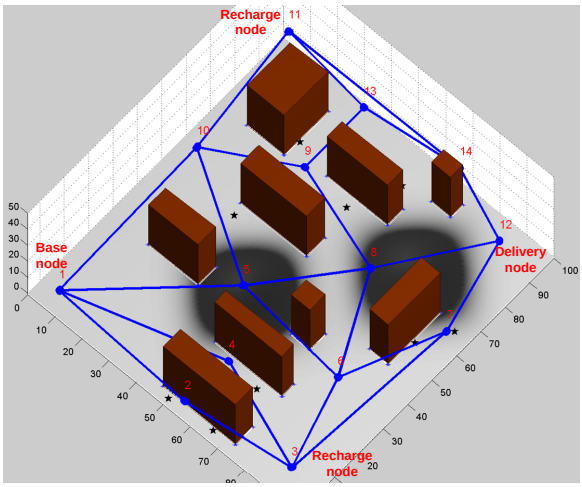

Figure 4: The simulation environment with obstacles (brown), PRM graph (blue), landmarks (stars), and windy areas (grey regions).

and

$$m = \frac{1}{1 + e^{-0.5sw}} - \frac{1}{1 + e^{0.5sw}} \tag{14}$$

For the given vehicle, fuel, and health dynamics, we apply the proposed health-aware method with search depth $H = 4$ and compare it with a pure reactive method, over a range of wind intensities by varying the parameter η. In the considered pure reactive method, the robot goes toward the closest recharge station when its fuel or health drops below the 10% of the nominal (maximum) fuel or health. Otherwise it goes toward the base or delivery depending on his package state value.

The second comparison aims to shed light on what happens if we ignore the exploited structure in this paper and propagate the joint belief of vehicle-level and mission-level states. In that case, one needs to compute the vehicle-level costs and constraints (collision probabilities) online which can be arbitrarily expensive depending on the environment. Moreover, since the dimension increases, the number of Monte Carlo samples should significantly increases to achieve the same accuracy in approximating the probability of constraint violation. This increase in computation limits us to shorter search horizons which restricts the applicability of these methods in scenarios. Although such the extra computation can be arbitrarily high, we run our algorithm on a shorter horizon $H = 2$ and compare it with the $H = 4$ to show the effect of horizon on the performance.

Figures 5 and 6 show the mean and variance of the number of delivered packages and the number of failures for different values of η. For each η, we perform 20 Monte Carlo runs to compute the mean and variance, where each run lasts for 2000 steps (on average, traversing each edge takes about 30 time steps). As it can be seen in these figures, increasing the wind intensity the failure rate for the pure-reactive method significantly increases, whereas the health-aware planner can take the stochasticity of health and its measurement into account and provide more robust plans.

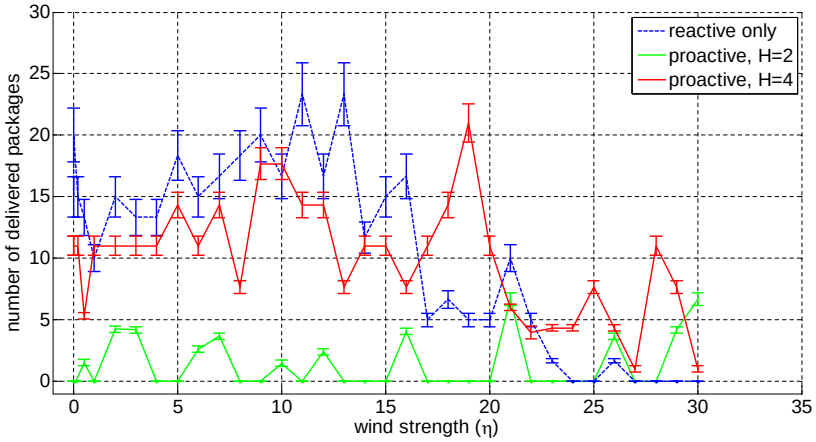

Figure 5: Number of delivered packages at different wind noise intensities

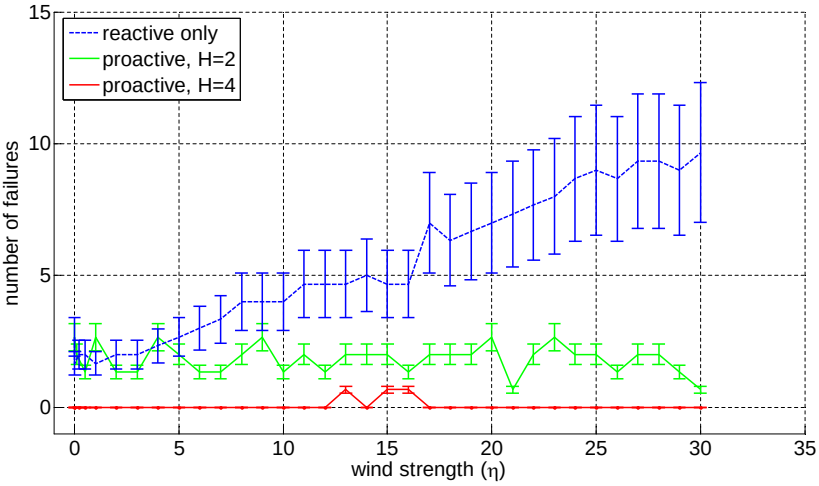

Figure 6: Number of failures at different wind noise intensities

5 Conclusion

This work developed a health-aware planning algorithm for persistent mission over partially observable domains. The algorithm works on a belief space that is decomposed into vehicle-level dynamics (stabilizable part) and mission-level dynamics (non-stabilizable part, such as health, fuel, etc.) parts and correspondingly applies graph based methods and forward search to each of these parts to generate a policy. Due to computational complexity reductions that result from this decomposition, the algorithm can generate proactive plans over larger-scale domains as well as real-time replanning capabilities that can't be handled by existing belief space planning approaches. The performance of the algorithm is demonstrated on a quadrotor persistent package deliver mission with stochastic battery and actuator degradation dynamics. It is shown that, compared to an alternative approach that ignores the health dynamics, the devel-

oped approach leads to higher number of delivered packages and reduced number of actuator failures.

Acknowledgments

This research was generously supported by Boeing Research & Technology in Seattle, WA.

References

[1] I. K. N. Nigam, S. Bieniawski and J. Vian, "Control of multiple uavs for persistent surveillance: Algorithm and flight test results," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, pp. 1236–1251, September 2012.

[2] S. Banker, "Amazon and drones – here is why it will work," Dec. 2013. [Online]. Available: http://www.forbes.com/sites/stevebanker/2013/12/19/amazon-drones-here-is-why-it-will-work/

[3] "Wildlife conversation uav challenge," Online http://www.wcuavc.com/, 2013.

[4] "Aerovironment," Online https://www.avinc.com/public-safety/applications/oilandgas, 2013.

[5] "Drones: The next firefighting tool," Online http://gazette.com/drones-the-next-firefighting-tool/article/1504041, 2013.

[6] J.-S. Marier, C. A. Rabbath, and N. Lechevin, "Health-aware coverage control with application to a team of small uavs," *Control Systems Technology, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2012.

[7] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley & Sons, Inc., 1994.

[8] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable markov processes over a finite horizon," *Operations Research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[9] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[10] N. K. Ure, G. Chowdhary, J. P. How, M. Vavarina, and J. Vian, "Health aware planning under uncertainty for uav missions with heterogeneous teams," in *Proceedings of the European Control Conference*, Zurich, Switzerland, July 2013.

[11] L. F. Bertuccelli and J. P. How, "Robust UAV search for environments with imprecise probability maps." 2005.

[12] R. Bellman, "Dynamic programming and stochastic control processes," *Information and Control*, vol. 1, no. 3, pp. 228–239, 1958.

[13] L. Buşoniu, R. Babuška, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 38, no. 2, pp. 156–172, 2008.

[14] W. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. Wiley-Interscience, 2007, pp. 225–262.

[15] S. Thrun, *Robotic mapping: A survey*. Morgan Kaufmann, 2002.

[16] K. Murphy, "A survey of POMDP solution techniques," 2000.

[17] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online planning algorithms for POMDPs," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[18] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *International Journal of Robotics Research*, vol. 28, no. 11-12, October 2009.

[19] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 1448–1465, 2011.

[20] D. Grady, M. Moll, and L. E. Kavraki, "Automated model approximation for robotic navigation with POMDPs," in *ICRA*, 2013.

[21] H. Kurniawati, T. Bandyopadhyay, and N. Patrikalakis, "Global motion planning under uncertain motion, sensing, and environment map," *Autonomous Robots*, pp. 1–18, 2012.

[22] A. Agha-mohammadi, S. Chakravorty, and N. Amato, "FIRM: Feedback controller-based Information-state RoadMap -a framework for motion planning under uncertainty-," in *International Conference on Intelligent Robots and Systems (IROS)*, SanFrancisco, 2011.

[23] A.-a. Agha-mohammadi, S. Chakravorty, and N. Amato, "Firm: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements," *International Journal of Robotics Research (IJRR)*, vol. 33, no. 2, pp. 268–304, 2014.

[24] A. Geramifard, T. J. Walsh, S. Tellex, G. Chowdhary, N. Roy, and J. P. How, "A tutorial on linear function approximators for dynamic programming and reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 6, no. 4, pp. 375–451, 2013.

[25] G. Chowdhary, T. Yucelen, M. Mühlegg, and E. N. Johnson, "Concurrent learning adaptive control of linear systems with exponentially convergent bounds," *International Journal of Adaptive Control and Signal Processing*, 2012.

[26] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003, pp. 1025–1032.

[27] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observatoins," in *Proceedings of Robotics: Science and Systems (RSS)*, June 2010.

[28] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, April 2008.

[29] B. Bole, C. Teubert, C. C. Quach, E. Hogge, S. Vazquez, K. Goebel, and G. Vachtsevanos, "Sil/hil replication of electric aircraft powertrain dynamics and inner-loop control for v&v of system health management routines," in *Annual conference of the prognostics and health management society*, 2013.

[30] J. Ghidella and P. Mosterman, "Requirements-based testing in aircraft control design," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2005.

6 Appendix

6.1 Quadrotor State

The stabilizable states (see Section 2) x^s consists of $(\mathtt{p}_x, \mathtt{p}_y, \mathtt{p}_z, \dot{\mathtt{p}}_x, \dot{\mathtt{p}}_y, \dot{\mathtt{p}}_z, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi})$, where $\mathtt{p} = (\mathtt{p}_x, \mathtt{p}_y, \mathtt{p}_z) \in \mathbb{R}^3$ denotes the position of the quadrotor and $(\phi, \theta, \psi) \in [-\pi, \pi]^3$ denotes the orientation of the quadrotor.

R is an orthonormal rotation matrix equivalent to the specified Euler angles. These correspond to rotations about the Z, Y, Z axes, respectively. Accordingly, we convert Euler angles to the corresponding rotation matrix as:

$$R = R_z(\phi)R_y(\theta)R_z(\psi) \tag{15}$$

6.2 Quadrotor dynamics

The continuous dynamics are:

$$\dot{x} = f(x, u, w) = \Bigg(\dot{\mathtt{p}}_x, \dot{\mathtt{p}}_y, \dot{\mathtt{p}}_z, g\phi, -g\theta, \frac{1}{m}(\hat{\delta}_{collective} + w_1),$$
$$\dot{\phi}, \dot{\theta}, \dot{\psi}, \frac{L}{I_x}(\delta_{roll} + w_2), \frac{L}{I_y}(\delta_{pitch} + w_3), \frac{1}{I_z}(\delta_{yaw} + w_4)\Bigg)^T + w_{5:16}, \tag{16}$$

which can be written as

$$\dot{x} = A_c x + B_c u + G_c w, \;\; w \sim \mathcal{N}(0, Q) \tag{17}$$

where

$$A_c = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{18}$$

$$B_c = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ m^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{L}{I_x} & 0 & 0 \\ 0 & 0 & \frac{L}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{pmatrix}, \;\; G_c = [B_c, I_{12}] \tag{19}$$

Therefore, the discrete version is:

$$\begin{aligned} x_{k+1} &= x_k + A_c \delta t x_k + B_c \delta t u_k + \sqrt{\delta t} G_c w_k, \;\; w_k \sim \mathcal{N}(0, Q) && (20) \\ &= A x_k + B u_k + G w_k, \;\; w_k \sim \mathcal{N}(0, Q) && (21) \end{aligned}$$

where

$$A = I_{12} + A_c \delta t, \;\; B = B_c \delta t, \;\; G = G_c \sqrt{\delta t} \tag{22}$$

6.3 Wind model

An any given position of the vehicle x^s, wind magnifies the intensity of the noise by teh value of the function $\mathcal{W}(\cdot)$ evaluated at x^s. For a one dimensional wind, where the wind center is denoted by c and the wind width is denoted by w, $\mathcal{W}$ is computed via the following bell-shaped function modeling the wind whose decay slope is specified by parameter s:

$$\mathcal{W}(x) = m^{-1} \left(\frac{1}{1 + e^{-s(x - c_l)}} - \frac{1}{1 + e^{-s(x - c_r)}} \right) \tag{23}$$

where

$$c_l = c - 0.5w, \;\; c_r = c + 0.5w \tag{24}$$

and

$$m = \frac{1}{1 + e^{-0.5sw}} - \frac{1}{1 + e^{0.5sw}} \tag{25}$$

Recall that the fuel consumption uncertainty is modeled as a random variable $w^v \sim \mathcal{N}(0, Q^v)$.

Accordingly,

$$Q^v = (Q^v_{bias} + \eta \mathcal{W}(x))^2. \tag{26}$$

6.4 Open-loop solver

Given the quadrotor kinematics, for a given start x_{start} and goal configuration x_{goal}, a straight line connecting start to goal is given by the sequence $(x_0, u_0, x_1, u_1, \cdots, x_{T-1}, u_{T-1}, x_T)$, where

$$x_0 = x_{start}, \;\; x_T = x_{goal}, \;\; x_{k+1} = f(x_k, u_k, 0), \;\; \forall k = 0, 1, \cdots, T \qquad (27)$$