SM2P: Towards a Robust Co-Pilot System for Helicopter EMS

Ian Mallett[1], Marcus Hoerger[1], Surabhi Gupta[2], Nisal Jayalath[2], Felipe Trevizan[1], Andrew Hunt[2], Hanna Kurniawati[1], Christophe Guettier[3]

[1]School of Computing, Australian National University
{hanna.kurniawati, felipe.trevizan}@anu.edu.au ; {ian.p.mallett, hoergems}@gmail.com
[2]Safran Electronics and Defense Australasia
{surabhi.gupta, andrew.hunt}@safrangroup.com ; nisaljayalath@gmail.com
[3]Safran Electronics and Defense
{christophe.guettier}@safrangroup.com

Abstract

This paper presents our preliminary work towards developing robust decision-making components for an automated co-pilot system in Helicopter Emergency Medical Services (HEMS). Specifically, in this paper, we focus on the integrated mission-motion planning framework for such a mission, and propose Stochastic-based Mission-Motion Planner (SM2P). SM2P frames the mission planning as a Stochastic Shortest Path (SSP), and the trajectory planning as a Partially Observable Markov Decision Processes (POMDPs). Each planning problem is solved using state-of-the art (approximate) solvers that can (re-)compute a good strategy that accounts for the various uncertainty plaguing an HEMS mission, on-line, within seconds for a typical HEMS mission. The use of SSP in mission planning allows SM2P to account for the non-deterministic effects of actions due to problem abstraction and limited condition in which the HEMS mission often takes place, while the use of POMDP allows SM2P to account for both the non-deterministic and partially observable nature of the operation as more information are perceived. Preliminary results on a simulation of three different HEMS scenarios in Corsica region indicates that SM2P reaches a success rate of over 95% in all scenarios.

Introduction

Helicopter Emergency Medical Services (HEMS) is among the most challenging and dangerous air operations (Hart March 2017). Its accident rate is more than 28 times higher than that of commercial aircraft (Holland and Cooksley 2005). HEMS missions are time-critical and arrive with little to no forewarning, making extensive planning difficult. These conditions are known to increase the chance of catastrophic mistakes (nsa October 2013). Despite these difficulties, most HEMS are performed without a co-pilot, which means a single person —the pilot— is wholly responsible for: guidance and navigation, detecting hazards to avoid, evaluating the suitability of landing zones and the ability to takeoff after landing, maintaining situational and spatial awareness of the terrain and proximate obstacles during manoeuvres in ground proximity, planning and re-planning routes given weather or mission parameter changes, as well as flying the helicopter. To help reduce the pilot's burden, in this paper, we present our preliminary work in developing a core planning component of an AI-based co-pilot for Helicopter Emergency Medical Services (HEMS).

Pilot Assistance Systems for helicopters, including autopilot, exist. For instance, DLR and ONERA, together with Eurocopter and Airbus, have developed multiple Pilot Assistance Systems (Lantzch et al. 2012; Le Blaye 2003; Lüken and Korn 2007). However, they are not suitable for HEMS, as they do not account for uncertainty and terrain characteristics, which are important if they were to provide manoeuvring strategies with ground proximity. Recently, Choudhury, et.al. have proposed a full-scale autonomous-flight system for HEMS missions (Choudhury et al. 2019). They identify uncertainty as a critical issue in HEMS mission, but to keep computation cost low, they address uncertainty by inflating risks: Deterministic planning is performed in a model of the world where risks have been inflated (e.g., obstacles have been enlarged) and assumes that no uncertainty remains. This approach works well when the feasible solution space is large. But, when the set of feasible solutions is small, inflating risks may remove all possible solutions, causing the mission to be deemed infeasible. Unfortunately, such scenarios are common in HEMS. For instance, when the patients to be picked up is in a bushfire area or mountainous areas —bushfires are frequent in many parts of the world, including in Australia, California, the Borneo Island in Indonesia, and South of France.

To alleviate the above difficulties, we propose to develop an AI-based co-pilot system that suggests good decisions within a limited computation time, while accounting for imperfect information about the mission and operating environment. Specifically, we propose an integrated mission and trajectory planner, where the mission planner is modelled as a Stochastic Shortest Path (SSP) and the trajectory planner is modelled as a Partially Observable Markov Decision Processes (POMDPs) problem. We call this approach Stochastic-based Mission-Motion Planner (SM2P).

The mission planner in SM2P abstracts the helicopter's operation area. It receives information from mission command and makes high-level strategy to accomplish the given mission. An example of this strategy is which victim to pick-

up first and the areas the helicopter needs to fly to in order to pick-up the victim. The trajectory planner receives a high-level guide from the mission planner and information about the environment from the helicopter's sensors and the pilot. It computes a motion strategy (e.g., collision-free trajectories, selection on landing area, etc.). This motion strategy is provided to the pilot. SM2P assumes the pilot will execute the provided strategy with some potential deviations from the suggested plan.

The mission and trajectory planners of SM2P communicates closely. When the mission planner receives an update on the mission, which require changes to its original mission plan, it will inform the trajectory planner, which will in turn update its motion strategy. Similarly, when the trajectory planner finds a certain guide is not feasible, it informs the mission planner, which in turn will find an alternative high-level strategy.

Preliminary results on three different HEMS scenarios in Corsica region near St Florent and Bastia indicate SM2P has a success rate of over 95% with 10 minutes offline computation per region, which can be computed once for HEMS missions in the region, 40 seconds offline computation per HEMS mission, and 5 seconds online computation.

Background and Related Work

Background on SSPs

A **Stochastic Shortest Path problem** (SSP) (Bertsekas and Tsitsiklis 1991) is a tuple $\mathbb{S} = \langle S, s_0, G, A, P, C\rangle$ in which: S is the finite set of states; $s_0 \in S$ is the initial state; $G \subseteq S$ is the non-empty set of goal states; A is the finite set of actions; $P(s'|s,a)$ is the probability of reaching s' after action a is applied in state s; and $C(s,a) \in \mathbb{R}_{>0}$ is the immediate cost of applying action a in state s. For simplicity, we assume $s_0 \notin G$ and we represent by $A(s)$ the actions applicable in state s.

A solution to an SSP is a *policy* π, i.e., a mapping from states to actions, and the optimal solution is any policy π^* that (i) reaches G from s_0 with probability 1; and (ii) has minimum total expected cost of reaching the goal from s_0. where the expected cost of a policy satisfying (i) can be computed using the following set of equations:

$$V_\pi(s) = C(s,\pi(s)) + \textstyle\sum_{s'\in S} P(s'|s,\pi(s))V^*(s')$$

for all $s \in S \setminus G$ and $V_\pi(s_g) = 0$ for all $s_g \in G$.

In this work, we consider SSPs with dead ends, that is, we do not assume that, for all $s \in S$, the probability for reaching G from s is 1. We use the fixed-cost approach for dead ends, i.e., if s is a dead end, then $V_\pi(s)$ is defined as d, where d is a large positive penalty for not reaching the goal. This approach allows us to transform the original SSP $\mathbb{S}$ into a new SSP $\mathbb{S}$' without dead ends in which there is an action that deterministically transitions from any state s to G with cost d (Mausam and Kolobov 2012). Notice that our approach can be trivially applied to different approaches to handling dead ends that optimizes different metrics relating cost and probability of reaching dead ends (e.g., see (Trevizan, Teichteil-Königsbuch, and Thiébaux 2017)).

Background on POMDP

Formally a POMDP is a tuple $\langle S, A, O, T, Z, R, \gamma\rangle$, where S, A and O are the state, action and observation spaces of the robot. T and Z model the uncertainty in the effect of taking actions and receiving observations as conditional probability functions $T(s,a,s') = p(s'|s,a)$ and $Z(s',a,o) = p(o|s',a)$, where $s, s' \in S$, $a \in A$ and $o \in O$. $R(s,a)$ models the reward the robot receives when performing action a from s and $0 < \gamma < 1$ is a discount factor. Due to uncertainties in the effect of performing actions and receiving observations, the true state of the robot is only partially observable. Hence, instead of planning with respect to states, the robot plans with respect to probability distributions $b \in \mathcal{B}$ over the state space, called beliefs, where $\mathcal{B}$ is the set of all probability distributions over S. The solution of a POMDP is an optimal policy π^*, a mapping from beliefs to actions $\pi^* : b \mapsto a$ such that the robot maximises the expected discounted future reward when following π^*. Once π^* has been computed, it can be used as a feedback-controller: Given the current belief b, the robot performs $\pi^*(b)$, receives an observation $o \in O$ and updates its belief according to $b' = \tau(b,a,o)$, where τ is the Bayesian belief update function. The value achieved by a policy π at a particular belief b can be expressed as

$$V_\pi(b) = R(b,\pi(b)) + \gamma \int_{o\in O} Z(b,\pi(b),o) V_\pi(\tau(b,\pi(b),o))\mathrm{d}o \tag{1}$$

where $R(b,a) = \int_{s\in S} R(s,a)b(s)\mathrm{d}s$ and $Z(b,a,o) = \int_{s'\in S} Z(s',a,o)\int_{s\in S} T(s,a,s')b(s)\mathrm{d}s\mathrm{d}s'$. The optimal policy π^* is then the policy that satisfies $\pi^*(b) = \mathrm{argmax}_\pi V_\pi(b)$.

Overall Framework

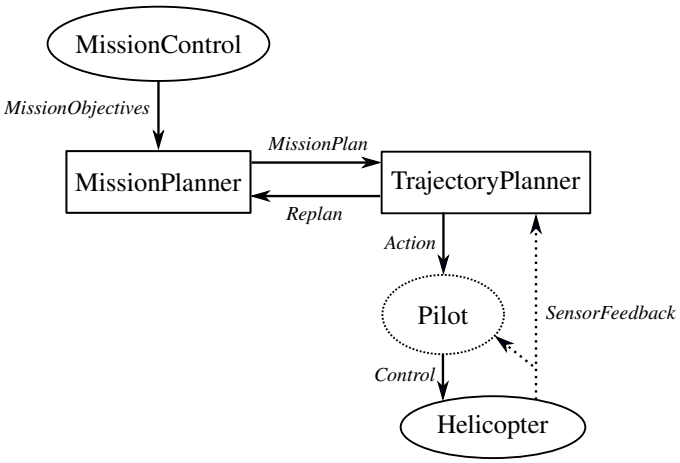

Figure 1: Overview of the planning architecture.

Most HEMS missions require the co-pilot system to solve a large and complex planning problem in three senses: Large hybrid state space, long planning horizon, and complex dynamics. The planner must operate on both continuous and

discrete variables, as it must identify which victims to pick-up and in which order, which medical facility to go to, so as to maximise the number of victims being saved, and navigation guidance in confined and imperfectly known areas that respect fuel and cloud ceiling requirements. The problem requires a long planning horizon. For instance, in the HEMS scenarios around Saint Florent and Bastia, for a helicopter to move from its starting position to the nearest hospital at average speed, it needs at least 60 planning steps. When the helicopter needs to pick-up a victim first, this planning step will substantially increase. Last but not least, helicopter dynamics is known to be complex. In this paper, we focus on the first two issues, and simplify the helicopter dynamics as a second-order discrete-time stochastic model according to:

$$f(\phi_t, \lambda_t, \theta_t, h_t, \nu_t) = \begin{bmatrix} \phi_t + \Delta\nu_t \cos\theta_t \\ \lambda_t + \Delta\nu_t \sin\theta_t \\ h_t + \Delta(\delta_h + e_h) \\ \theta_t + \Delta(\delta_\theta + e_\theta) \\ \nu_t + \Delta(\alpha + e_\nu) \end{bmatrix} \quad (2)$$

where $\phi_t, \lambda_t, \theta_t, h_t, \nu_t \in \mathbb{R}$ are the latitude, longitude, yaw-orientation, elevation and forward velocity of the helicopter. δ_h and δ_θ are fixed climb and turn rates, whereas α is a fixed acceleration. Δ is a fixed control duration and e_h, e_θ, e_ν represent random control errors that are drawn from zero-mean Gaussian distributions.

Figure 1 presents the overall framework of our integrated mission and trajectory planners. The mission planner computes the high level strategy on which victims to pick-up and which order should they be picked-up. These strategies are transformed into navigation guides that respect the estimated fuel usage and flying requirements under the estimated weather conditions. These strategies are computed, while respecting fuel and weather condition requirements.

To construct navigation guides, the mission planner constructs an abstraction of the operation area of HEMS. Suppose $\mathcal{W} \subseteq \mathbb{R}^3$ is the bounded operational space of the HEMS mission. The space $\mathcal{W}$ contains the terrain of the area. The mission planner uses the SPArse Roadmap Spanner (Dobson, Krontiris, and Bekris 2013), to construct a sparse graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ that captures the connectivity of $\mathcal{W}$ well. Each vertex in $\mathcal{V}$ is associated with a position sampled from $\mathcal{W}$, while an edge $\overline{vv'} \in \mathcal{E}$ means there is a collision-free straight line-path for the helicopter from the positions represented by $v \in \mathcal{V}$ to $v' \in \mathcal{V}$. The environment $\mathcal{W}$ is then decomposed into Voronoi regions, where the points are the positions associated with the vertices $\mathcal{V}$. The navigation guide provided by the mission planner to the trajectory planner is then a mapping from one Voronoi region to the neighbouring Voronoi region to visit, so as to achieve the mission objective.

The trajectory planner is an on-line POMDP planner and receives observations about the environment around its current location from the helicopter's sensors and from the pilot. Such observations will enable the trajectory planner to realise if moving into a certain Voronoi region has now become very difficult or even infeasible. For example, if a no-fly zone were to suddenly appear due to the spread of fire in bush-fire areas, the navigation guide from the mission planner would be rendered invalid. When this situation happens, the trajectory planner notifies the mission planner. The mission planner will identify the affected components of $\mathcal{G}$, modify the graph, and re-plans with respect to the modified graph. The new navigation guide is then passed to the trajectory planner.

The subsequent sections provide details of the mission and trajectory planners.

Mission Planner Details

The mission planner provides a high level policy to guide the trajectory planner to achieve its mission objectives, primarily guiding it to the additional victim to be rescued, and to the hospital. The policy is made with respect the helicopter's position, current fuel level, and the height of an overcast cloud ceiling. Under Visual Flight Rules conditions (VFR), the helicopter must not cross this ceiling unless absolutely necessary. In the experiement, we follow the VFR rules followed in France.

We assume that, in the case where there are one or more victims to pick up en route, that there is space for only one more victim on board, and that when arriving to pick up a victim, it may turn out that it is impossible to get the victim on board. Given this, the mission planner's goal is to either successfully pick up a victim, or to try to pick them all up.

We use the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ obtained using SPArse Roadmap Spanner (Dobson, Krontiris, and Bekris 2013) to define the SSP $\langle S_{\text{MS}}, s_0, G, A_{\text{MS}}, P, C\rangle$ where:

1. The mission planner state space S_{MS} is the product of $\mathcal{V}$ with a discretised set of fuel levels F and discretised cloud heights H_c. In the case where there are one or more victims to pick up en route (i.e., $|\Upsilon| > 1$), we include a variable $\texttt{at-capacity} \in \{\top, \bot\}$, and for each victim $\nu \in \Upsilon$, we include $\texttt{attempted}_\nu \in \{\top, \bot\}$, representing whether the pilot has tried to pick up that victim.
2. The initial state $s_0 \in S_{\text{MS}}$ is the initial position of the helicopter, fuel level equivalent to a full tank, and an initial cloud height.
3. $G \subset S_{\text{MS}}$ is the set of states where the helicopter is at the hospital and, when there are extra victims in the scenario, either an extra victim is on board ($\texttt{at-capacity} = \top$) or $\forall\nu, \texttt{attempted}_\nu = \top$.
4. The set of possible actions A_{MS} is $\{\texttt{move}(v, v') \mid \overline{vv'} \in \mathcal{E}\} \cup \{\texttt{pick-up}_\nu \mid \nu \in \Upsilon\}$. Let $v_s \in \mathcal{V}$ be the helicopter's location in state $s \in S_{\text{MS}}$. If $\texttt{at-capacity}$ equals $\top$ in s, then the set of applicable actions $A_{\text{MS}}(s)$ is $\{\texttt{move}(v_s, v') \mid \overline{v_s v'} \in \mathcal{E}\}$, otherwise $A_{\text{MS}}(s)$ also includes $\{\texttt{pick-up}_\nu | \nu \in \Upsilon, v_s \text{ is the closest vertex to } \nu\}$.
5. The probability transition function $P(s'|s, a)$ is such that each component of s is changed *independently* as follows:
   - **Location**: if $a = \texttt{move}(v_s, v')$, then the location at the new state s' is v' with probability 0.9 (i.e., $P(v_{s'} = v') = 0.9$) and, with probability 0.1, the location does not change (i.e., $P(v_{s'} = v_s) = 0.1$). The location remains the same for all non moving actions.

- **Fuel level**: for $a = \texttt{move}(v_s, v')$, the fuel level decreases deterministically, assuming the helicopter moves from v_s to v' at 150 km/h, using fuel at a steady rate of 4kg per minute. For $\texttt{pick-up}_\nu$, the fuel decreases by 20kg on a success, or 4kg on a failure (we assume the pickup takes 5 or 1 minutes).
- **Cloud ceiling**: let c_s and $c_{s'}$ be the cloud ceiling heights in states s and s'. $P(c_{s'} = c_s) = 0.98$, $P(c_{s'} = c_s + 100m) = 0.01$ and $P(c_{s'} = c_s - 100m) = 0.01$. If $c_{s'}$ would be outside $[600m, 900m]$, it is set to the nearest bound.
- **Victims**: for $a = \texttt{move}(v_s, v')$, $\texttt{at-capacity}$ and $\texttt{attempted}_\nu$ are not changed. If $a = \texttt{pick-up}_\nu$, $P(\texttt{attempted}_\nu = \top) = 1$, $P(\texttt{at-capacity} = \top) = 0.95$, $P(\texttt{at-capacity} = \bot) = 0.05$.

6. The cost function represents the expected time it takes to complete an action, with a 15 minute penalty for entering the cloud layer. Moreover, $C(s, \texttt{pick-up}_\nu) = 288$, $C(s, \texttt{move}(v_s, v'))$ is the time in seconds to travel from v_s to v' at 150 km/h, plus 900 if v_s is above the current cloud height.

Our probability transition function $P(s'|s, a)$ defined above implies that moving from a location v to v' follows a geometric distribution with $p = 0.9$ since the action $\texttt{move}$ succeeds with probability 0.9 and fails by staying at the same location v with probability 0.1. While successive executions of the same $\texttt{move}$ action might not change the helicopter's location, it changes the current state $s \in S_{\text{MS}}$ of the system due to the deterministic fuel consumption (the fuel component of the state monotonically decreases) and potential change in the cloud ceiling.

Trajectory Planner Details

The trajectory planner's purpose is to compute a low-level motion strategy for the pilot to successfully complete the mission on-line, using the high-level mission-planner strategy as a navigation guide. During run-time the trajectory planner uses sensor information from the helicopter to construct and maintain a local map of the environment that includes the terrain, as well as obstacles (such as trees, power lines or buildings), cloud-ceiling and possible no-fly zones within a bounded region $\mathcal{W}_{local} \subset \mathcal{W}$ around the helicopter. This local map is updated after every step (in this paper we assume that the geometries and locations of the obstacles within $\mathcal{W}_{local}$ are perfectly known to the trajectory planner). The low-level motion strategy is then computed with respect to the current local map and the stochastic dynamics of the helicopter.

Due to imperfect controls and sensor information the true state of the helicopter is only partially observed. Thus, we maintain a belief $b_t \in b$ over the current state of the helicopter and formulate the problem of computing a low-level motion strategy for the pilot as a POMDP. To increase efficiency, we further decompose the problem into two sub-problems, both formulated as separate POMDPs: Navigating and landing/take-off. Details on the POMDP formulation of both sub-problems are provided in Sub-Sections **POMDP-Formulation of the Navigation Problem** and **POMDP-Formulation of the Landing Problem** respectively. For the first sub-problem, the trajectory planner computes a policy to navigate to the Voronoi regions in the environment as specified by the mission planner policy. For the second sub-problem, we first construct a set of possible landing zones across the environment off-line (details on how these landing zones are constructed are provided in Sub-Section **Constructing the Landing Zones**). At run-time, the trajectory planner then determines the closest landing zone to the victim and computes a motion strategy to safely touch-down at the landing zone in order to pick-up the victim. During run-time the trajectory planner switches between both POMDP problems depending on the current mission planner objective.

To compute a policy from the current belief $b_t \in \mathcal{B}$, we use ABT (Kurniawati and Yadav 2013), one of the fastest Monte-Carlo-Tree-Search based on-line solvers. A summary of this method is presented here for completeness. Starting from b_t (we represent beliefs as sets of particles) ABT approximates the optimal policy by constructing and evaluating a belief-tree, whose nodes represent beliefs and edges represent pairs of actions and observations. To construct the belief-tree, ABT samples *episodes*, that is, sequences of state-action-observation-reward quadruples, starting from the current belief and associates the states of the episode with nodes in the belief-tree. To select actions during the episode sampling process, ABT uses Upper Confidence Bounds1 (UCB1) (Auer, Cesa-Bianchi, and Fischer 2002) which ensures asymptotic convergence towards the optimal policy. The states, observations and reward of an episode are sampled from a generative model that encodes the transition, observation and reward functions. An episode is expanded until either a terminal state is reached, or the episode reaches a belief-node for which there are actions that haven't been visited before. If the first terminating condition occurs, ABT backpropagates the sampled reward-trajectory to update the estimates of $\widehat{Q}(b, a)$ for each belief $b \in \mathcal{B}$ associated to a state in the episode, where $Q(b, a)$ is the value of executing action $a \in A$ from b and continuing optimally afterwards. Otherwise, ABT first estimates the value of the last belief node by simulating a rollout strategy from the last state of the episode and then continues backpropagating the sampled reward-trajectory as described above.

Once the planning time for the current step is over, ABT selects an action according to $\pi(b_t) = \text{argmax}_{a \in A} \widehat{Q}(b_t, a)$. After executing the action and receiving an observation, the current belief is updated (we use a Sequential-Importance-Resampling (SIR) particle filter (Arulampalam et al. 2002) to update the belief) and planning continues from the updated belief.

To embed the navigation guides from the mission planner into the POMDP policy search, we construct a rollout strategy that encodes the mission planner strategy. This is done as follows: Suppose b_t is the current belief and $\bar{s} \in S$ is the mean state of b_t. Using $\bar{s}$, we estimate the Voronoi region – with associated vertex $v_1 \in \mathcal{V}$ – the helicopter is currently located in and query the mission planner policy.

This provides us with a high-level action that can either be $\texttt{move}(v_1, v_2)$, i.e. to navigate from the Voronoi region associated with v_1 to the Voronoi region associated with $v_2 \in \mathcal{V}$, or $\texttt{pick-up}_{v_1}$, i.e. to pick up the victim located in the current Voronoi region. In the first case, we compute a motion strategy to reach the Voronoi region associated to v_2, assuming deterministic effects of actions. In the second case we assume deterministic dynamics too, but compute a motion strategy to reach a landing zone near the victim.

This allows us to guide the search towards achieving the high-level strategy computed by the mission planner, while simultaneously planning with respect to the stochastic helicopter dynamics and the local environment around the helicopter. Note that we only query the mission planner after the current belief b_t has been updated. Within a planning step (that is, during the policy computation from b_t), the high-level action remains constant.

POMDP-Formulation of the Navigation Problem

State, Action and Observation Spaces The state space of the helicopter is defined as the cross-product of four components $S = \mathbb{R}^3 \times \Pi \times [0, \nu_{max}] \times \mathbb{R}^+ \times H_c$, where the first component is the 3D real-vector space consisting of the latitude, longitude and elevation of the helicopter above median sea level. $\Pi = [-180.0, 180.0]$ is the set of yaw-orientations (in degrees) of the helicopter, whereas $[0, \nu_{max}]$ are the minimum and maximum forward-velocities (in m/s) of the helicopter. The component $\mathbb{R}^+$ is the set of all fuel loads of the helicopter, whereas H_c is the discrete set of cloud-ceiling heights.

The action space of the helicopter is defined as $A = \{\texttt{accelerate}, \texttt{decelerate}, \texttt{climb}, \texttt{descend}, \texttt{turnLeft}, \texttt{turnRight}\}$. The `accelerate` and `decelerate` actions set the acceleration α in eq.(2) to a fixed positive/negative value. Similarly, the `climb`/`descend` and `turnLeft`/`turnRight` actions set the climb rate δ_h and turn rate δ_θ in eq.(2) to fixed positive/negative values respectively.

We assume that the helicopter is equipped with two types of sensors: A localization sensor which provides information regarding the current latitude, longitude and elevation of the helicopter and a gyroscope which provides information regarding the helicopter's yaw-orientation. More formally, the observation space is defined as $O = \mathbb{R}^3 \times \Pi$, where the first component describes the latitude, longitude and elevation of the helicopter Π is defined as above.

Transition Function To model the transition dynamics of the helicopter, we use the second-order stochastic dynamic system defined in eq.(2) given the latitude ϕ, longitude λ, elevation h and yaw-orientation θ associated to the current state of the helicopter. Additionally, we assume that the fuel load of the helicopter decreases deterministically by a constant rate (in our experiments, we assume a fuel consumption of $\frac{1}{3}kg$ per time step).

Observation Function The observation model of the helicopter is defined as

$$o_t = [\phi_t; \lambda_t; h_t; \theta_t]^T + e_o \tag{3}$$

where ϕ_t, λ_t, h_t, θ_t are the latitude, longitude, elevation an yaw angle components of the state. $e_o \in \mathbb{R}^4$ is a random vector drawn from a zero-mean multivariate Gaussian distribution representing sensor noise.

Reward function To encode the objective of reaching the hospital, the helicopter receives a reward of 10,000 when entering a goal area around the hospital (modelled as a sphere with radius 1,000m around the location of the hospital). Once the helicopter enters this goal area, the mission is considered successful. If the helicopter collides with the terrain or an obstacle, it receives a penalty of -50,000 and the mission is considered as unsuccessful. To encourage the helicopter to reach the goal as quickly as possible, it receives a penalty of -10 at every step. Since crossing the cloud ceiling is considered dangerous due to low visibility, the helicopter receives a penalty of -10 for every state where it is located above the cloud ceiling, discouraging the helicopter from crossing the cloud layer unless necessary (for instance to avoid an obstacle).

POMDP-Formulation of the Landing Problem

Once the trajectory planner receives a $\texttt{pick-up}_v$ action from the mission planner, it switches to the landing problem. For this problem the task is to safely navigate to a landing zone close to the injured person in the environment and perform a landing maneuver.

The POMDP formulation of this problem is similar to the one for the navigation problem described in the previous section, with some notable differences: We extend the action space of the helicopter with and additional `land` action whose purpose is to perform a vertical touch-down at the current location of the helicopter to pick-up the victim. For this action, the helicopter enters a terminal state and receives a reward of 10000 if the following conditions are met: a.) The helicopter is located above a landing zone area (the method to define and construct a landing zone is described in the next subsection), b.) The vertical distance between the helicopter and the terrain is within 75m and c.) The forward velocity of the helicopter is smaller than 10m/s. If at least one of these conditions is not satisfied, the helicopter enters a terminal state too, but receives a penalty of -50000 and the mission is considered unsuccessful. Note that we assume that in case the `land` action is successful, the injured person is automatically picked up and the trajectory planner switches back to the navigation problem.

Constructing the Landing zones In order for the helicopter to successfully pick up an injured person, it has to determine areas in the environment that are suitable for landing, that is, areas that are sufficiently large and flat. To construct such areas, we use a simple geometric approach: Suppose the terrain is represented by a triangular mesh, i.e. a set U of triangles. For each triangle $u \in U$, we compute its slope s_u via $s_u = \arccos(n_u \cdot z/(\|n_u\| \, \|z\|))$, where n_u is the normal vector of triangle u, $z = (0, 0, 1)^T$ and "$\cdot$" denotes the dot product. If s_u yields a value larger than a given threshold s_{max} (in our experiments we use $s_{max} = 9deg$), we remove the triangle u from U. In other words, we remove triangles from U that are too "steep" for the helicopter

to land on. We then incrementally merge the remaining triangles in U into subsets of triangles $\mathbb{U} = \{U_1, ...U_k\}$, with $U_i \subset U$ where for each triangle $u_1 \in U_i$, there is at least one $u_2 \in U_i$, $u_1 \neq u_2$ such that u_1 and u_2 share an edge in the original terrain mesh. Each $U_i \in \mathbb{U}$ is then a possible landing zone. However, some triangle sets in $\mathbb{U}$ might have an area (defined as the sum of the area of all triangles in the triangle set) that is too small for the helicopter to land on. To determine whether the triangles in a triangle set U_i provide a sufficient area for landing, we project the triangles in U_i onto the xy-plane and compute the area of the largest inscribed circle within the (possibly non-convex) boundary polygon of the projected triangles. If this area is smaller than a given threshold (in our experiments we use $75\pi m^2$), we remove U_i from $\mathbb{U}$. The remaining triangle sets in $\mathbb{U}$ are then the resulting landing zones.

Note that this process of constructing the landing zones is done off-line. During run-time, once the trajectory planner switches to the landing problem, we select the closest landing zone in $\mathbb{U}$ to the injured person (in terms of the distance of the location of the injured person to the geometric centers of the landing zones) which then becomes the target landing zone for the pilot to land on.

Parallelization of Belief Update and Policy Computation

Most on-line POMDP solvers (including ABT) typically follow a strictly sequential order of execution, that is, policy computation – policy execution – belief update. In practice, such an implementation would incur significant delays between the execution of two actions, due to the time required to update the current belief and compute a policy for the updated belief. To reduce these delays to a minimum, we parallelize policy computation, policy execution and parts of the belief update, similar to the method proposed in (Hoerger et al. 2019): While the helicopter executes an action $a \in A$, we run two processes in parallel.

The first process is the belief-update process which draws samples from a proposal distribution, (in our case $T(s, a, s')$) using state samples drawn from the current belief and the currently executed action. Once the helicopter receives an observation, all that remains for the belief update is to update the importance weights, up to a normalization constant, based on the perceived observation, which can be done fast.

The second process is the policy-update process. Once the helicopter starts executing a from the current belief b, our implementation of ABT plans for the next step by sampling additional episodes starting from the current belief, using the currently executed action as the first action of the sampled episodes, thereby improving the policy within the entire descendent of b via a in the belief tree. This strategy increases the chances that after the helicopter has executed a and the belief is updated based on the observation perceived, a good policy for the next belief is readily available.

Figure 2: The mission area used throughout the experiments. Initially the helicopter starts at Saint-Florent (shown as a blue marker) with a victim on board and the task is to navigate to the hospital in Bastia (shown as a green marker). The red markers indicate the locations of the additional victims the helicopter has to pick-up before continuing its mission to Bastia. *Image: Google Earth, earth.google.com/web/*

Experiments and Results

Problem Scenarios

To evaluate our system, we tested it on three problem scenarios in which a pilot operates on a map of size (28.1×26.0)km in the Corsica region in France near Saint-Florent and Bastia, shown in Figure 2. In all three scenarios the pilot starts near Saint-Florent at location $(42.68^0N, 9.302^0E)$ (shown as a blue marker in Figure 2) with an injured person on board of the helicopter and its task is to safely navigate to a hospital in Bastia at location $(42.740105^0N, 9.457107^0E)$ (shown as a green marker in Figure 2) to deliver the injured person. Furthermore, at the beginning of the mission, the helicopter is instructed to pick up an additional injured person in the environment before continuing its flight to Bastia. For the dynamic model of the helicopter defined in eq.(2) we assume that the constant acceleration α, climb rate δ_h and turn rate δ_θ is $\alpha = \pm 4m/s^2$, $\delta_h = \pm 6m/s$ and $\delta_\theta = \pm 4deg/s$ respectively. We further assume that the pilot applies each action for a control duration of $\Delta = 5s$.

For **Scenario 1**, the additional victim is located at $(42.667617^0N, 9.392734^0E)$ (shown as Victim #1 in Figure 2) and the pilot has to find a motion strategy to navigate to Victim #1, safely land near the additional victim to board it, and finally safely deliver both victims to the hospital in Bastia.

In **Scenario 2**, the pilot must pick-up Victim #1, too. However, at time $t = 15$, the pilot is informed about a no-fly zone near the current location of the helicopter. This no-fly zones causes the initial mission plan to become infeasible. Subsequently, the pilot must find an alternative strategy to reach Victim #1 before delivering both victims to the hospital.

The no-fly zone is simulated by modifying the mission

planner graph. Specifically, by removing the edge between vertices $v_1 \in \mathcal{V}$ and $v_2 \in \mathcal{V}$, where v_1 is the vertex associated to the Voronoi region the helicopter is located in at time $t = 15$ and v_2 is the vertex associated to the target Voronoi region of the current mission planner action $\texttt{move}(v_1, v_2)$.

For **Scenario 3** the pilot is initially informed that there's a victim at location $(42.732068^0N, 9.363066^0E)$ (shown as Victim #2 in Figure 2). At time $t = 20$ the mission planner receives an emergency call from Mission Control, informing it about a second victim at location $(42.688724^0N, 9.384275^0E)$ (shown as Victim #3 in Figure 2). However, since there's already a victim on board at the start of the mission, the helicopter has space for only one additional victim on board due to space and weight limits. Subsequently, the pilot must decide which of the two victims in the environment to pick up before continuing the mission to the hospital in Bastia.

Experimental Setup

For all three problem scenarios we first generated 10 graphs for the mission planner using the SPArse implementation provided by OMPL. Each graph was constructed by running SPArse for 10 minutes. We then compute, for each scenario and each graph a mission planner policy that serves as the initial mission plan for the trajectory planner.

For the mission planner we use Labelled SSiPP (Trevizan and Veloso 2012) with the Regrouped Operator Counting heuristic (Trevizan, Thiébaux, and Haslum 2017). The initial mission plan is computed off-line using one thread on an Intel Xeon Silver 4110 CPU with 2.1Ghz and 128GB of memory, and took approximately 13.6s (10s to construct the graph and 3.6s to compute the mission planner policy) on average per scenario per graph.

For the trajectory planner, all POMDP models as well as the parallelized version of ABT (as discussed in Sub-Section **Parallelization of Belief Update and Policy Computation**) were implemented in C++ within the OPPT-framework (Hoerger, Kurniawati, and Elfes 2018). For the POMDP models we used a discount factor of $\gamma = 0.98$. The size of the local map the trajectory planner maintained during runtime was set to be $4,000 \times 4,000$m. All simulations were run using 3 threads on an Intel Xeon Silver 4110 CPU with 2.1Ghz and 128GB of memory. We assume that each action the pilot executes takes 5s to complete, and provides a 5s planning time per step to the trajectory planner.

Results

For each problem scenario and each initial mission planner policy, we tested our system using 100 simulation runs. Table 1 shows the success rate, average number of steps and average total discounted reward achieved by SM2P in all three problem scenarios. In all three scenarios, SM2P achieved a success rate of at least 96% where the pilot successfully picked up a victim in the environment and reached the hospital, demonstrating the robustness of SM2P in challenging HEMS missions.

Table 2-5 provides detailed results for all Scenarios and mission planner graphs. These results indicate the robustness of SM2P. Recall that since the mission planner graphs are computed using a sampling-based method, these graphs are different between one run and another. Despite such differences, SM2P consistently achieve a success rate of over 94%.

Looking at the average number of steps for Scenario 1 and 2 in Table 1, we can see that it typically takes slightly longer (around 20 steps longer) to complete the mission in Scenario 2 compared to Scenario 1. This is not surprising. Due to the no-fly zone introduced at $t = 15$ in Scenario 2, the helicopter must take a slight detour to reach the victim. Furthermore, since the no-fly zone causes the initial mission plan to become invalid, it is paramount that the mission planner is able to quickly update its policy. On average, SM2P can update the mission plan during run-time in 5 seconds. This is roughly the same time it takes for the pilot to execute an action, which indicates that updating the mission planner policy is efficient enough for an on-line planning setting, even if there are structural changes to the mission planner graph (such as an edge being deleted).

Table 5 shows the number of times (in percent) per graph, SM2P decided to pick-up Victim #3 in Scenario 3. The decision of which victim to pick-up is affected by the Voronoi region the helicopter is estimated to be located in at time $t = 20$, when the mission planner is informed regarding the location of the additional Victim #3, causing the mission plan to be updated. For the majority of the runs and mission planner graphs, SM2P decided to pick up Victim #3, since the location of Victim #3 is closer to the hospital. A notable exception is graph 9 for which SM2P decided to pick up Victim #2 in 30% of the runs. The reason is that for graph 9, the helicopter often operates near the border of two neighbour Voronoi regions at time $t = 20$. Depending on which Voronoi region the helicopter is estimated to be located in, the mission planner policy suggests to navigate either to Victim #2 or Victim #3. Recall from Sub-Section **Trajectory Planner Details** that the current Voronoi region is estimated from the mean state of the current belief. The consistency of selecting which victim to pick up can be improved, for instance, by estimating the Voronoi region that contains the largest amount of probability mass of the current belief.

	Success Rate	Avg. num steps	Avg. total discounted reward
Scenario 1	96.8%	197.4 $\pm$ 2.1	-385.8 $\pm$ 56.2
Scenario 2	96.9%	217.1 $\pm$ 3.7	-468.2 $\pm$ 52.3
Scenario 3	97.6%	206.8 $\pm$ 2.3	-416.6 $\pm$ 55.6

Table 1: The success rate (in percent), Average number of steps and Average total discounted reward achieved by SM2P in all three problem scenarios. The average is taken over all 10 mission planner graphs using 100 simulation runs per scenario and graph. $\pm$ indicates the 95% confidence intervals.

Summary

This paper presents our preliminary work in developing a robust decision-making component to help a pilot performing

Graph	Success Rate	Avg. num steps	Avg. total discounted reward
1	94%	191.02 ± 8.8	-618.67 ± 183.54
2	95%	221.81 ± 6.3	-583.94 ± 230.53
3	100%	218.5 ± 3.8	-146.58 ± 109.75
4	97%	194.09 ± 4.4	-361.88 ± 312.16
5	99%	177.76 ± 4.4	-128.89 ± 190.0
6	98%	192.19 ± 9.8	-225.26 ± 377.9
7	94%	181.60 ± 6.0	-589.94 ± 228.06
8	95%	224.0 ± 5.5	-591.90 ± 155.67
9	100%	176.02 ± 6.3	-93.37 ± 90.39
10	96%	201.32 ± 6.9	-308.08 ± 196.99

Table 2: Results for **Scenario 1**. The success rate, average number of steps and average total discounted rewards are conputed over 100 simulation runs per mission planner graph. ± indicates the 95% confidence intervals.

Graph	Success Rate	Avg. num steps	Avg. total discounted reward
1	100%	207.83 ± 6.8	-46.49 ± 93.99
2	94%	225.72 ± 8.47	-618.64 ± 748.15
3	95%	220.52 ± 10.5	-525.93 ± 125.34
4	97%	218.61 ± 9.5	-638.31 ± 102.78
5	98%	206.82 ± 4.5	-167.16 ± 108.46
6	95%	192.55 ± 9.7	-618.4 ± 265.65
7	98%	195.9 ± 6.8	-452.21 ± 197.56
8	95%	218.36 ± 6.61	-663.73 ± 303.60
9	98%	211.09 ± 8.22	-530.31 ± 683.77
10	99%	205.77 ± 4.6	-167.32 ± 107.25

Table 3: Results for **Scenario 2**. The success rate, average number of steps and average total discounted rewards are conputed over 100 simulation runs per mission planner graph. ± indicates the 95% confidence intervals.

a HEMS mission —time-critical missions that are typically plagued by various types of uncertainty. In this work, we propose Stochastic-based Mission-Motion Planner (SM2P) as a framework that enable us to quantify uncertainty and account such a quantification in its decision-making. SM2P uses a combination of existing SSP and POMDP solvers that are tightly coupled by their ability to revise and recompute plans on-line within a few seconds. Simulation results indicate that SM2P is sufficiently efficient to compute good strategies for a co-pilot system in HEMS missions.

Future work abounds. For instance, in this work, we use a simplified helicopter model. Using high fidelity helicopter model will slow down the POMDP solver. Therefore, incorporating solvers that can perform well for problems with complex dynamics, such as (Hoerger et al. 2019) would be useful. Furthermore, uncertainty due to pilot stress level and experience have not been taken into account in this work. However, a reliable co-pilot system would account for such factors. Another dimension is handling potential degradation of the pilot's visibility. Although sensors that can "see

Graph	Success Rate	Avg. num steps	Avg. total discounted reward
1	100%	210.9 ± 7.4	-118.32 ± 116.30
2	97%	204.83 ± 6.1	-928.58 ± 458.79
3	96%	213.56 ± 8.2	-397.28 ± 396.25
4	100%	216.28 ± 7.4	-189.51 ± 173.52
5	98%	196.11 ± 7.4	-253.78 ± 386.41
6	96%	191.79 ± 8.2	-587.23 ± 195.28
7	98%	213.56 ± 8.9	-321.42 ± 164.33
8	96%	227.56 ± 6.8	-634.76 ± 587.32
9	100%	215.81 ± 5.4	-104.11 ± 176.50
10	95%	201.73 ± 5.4	-781.69 ± 426.47

Table 4: Results for **Scenario 3**. The success rate, average number of steps and average total discounted rewards are conputed over 100 simulation runs per mission planner graph. ± indicates the 95% confidence intervals.

Graph	Victim #3 pickup (in %)
1	93%
2	90%
3	100%
4	100%
5	87%
6	100%
7	100%
8	94%
9	70%
10	100%

Table 5: The number of times (in percent) SM2P decided to pick-up Victim #3 in **Scenario 3** for each mission planner graph over all simulation runs.

through" fogs have been developed, uncertainty due to degraded visibility conditions when operating in certain conditions (e.g., bushfires) and its effects to the pilot's capability will need to be considered in the recommendations provided by a co-pilot system. Last but not least, experiments on high fidelity simulator or physical system is needed for better validation.

Acknowledgments

We thank Colonel Andjy Zouag for the many discussions in developing realistic EMS scenarios used in the experiments for this paper.
This project is sponsored by Safran Electronics and Defense and Safran Electronics and Defense Australasia.

References

October 2013. National Search Rescue (NSARA). Helicopter Rescue Techniques. https://www.dco.uscg.mil/Portals/9/CG-5R/nsarc/Helicopter_Rescue_Techniques_NSARA_Manual_10-23-2013.pdf.

Arulampalam, M. S.; Maskell, S.; Gordon, N.; and Clapp, T. 2002. A tutorial on particle filters for online nonlinear/non-

Gaussian Bayesian tracking. *IEEE Transactions on signal processing* 50(2): 174–188.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47(2-3): 235–256. ISSN 0885-6125.

Bertsekas, D.; and Tsitsiklis, J. 1991. An Analysis of Stochastic Shortest Path Problems. *Mathematics of Operations Research* 16(3): 580–595. ISSN 0364-765X.

Choudhury, S.; Dugar, V.; Maeta, S.; MacAllister, B.; Arora, S.; Althoff, D.; and Scherer, S. 2019. High performance and safe flight of full-scale helicopters from takeoff to landing with an ensemble of planners. *Journal of Field Robotics* .

Dobson, A.; Krontiris, A.; and Bekris, K. E. 2013. Sparse roadmap spanners. In *Algorithmic Foundations of Robotics X*, 279–296. Springer.

Hart, C. A. March 2017. National Transportation Safety Board (NTSB). https://www.ntsb.gov/news/speeches/CHart/Documents/hart_20170302.pdf.

Hoerger, M.; Kurniawati, H.; and Elfes, A. 2018. A Software Framework for Planning Under Partial Observability. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1–9. IEEE.

Hoerger, M.; Song, J.; Kurniawati, H.; and Elfes, A. 2019. POMDP-based Candy Server: Lessons Learned from a Seven Day Demo. In *Proc. Int. Conference on Automated Planning and Scheduling (ICAPS)*.

Holland, J.; and Cooksley, D. G. 2005. Safety of helicopter aeromedical transport in Australia: a retrospective study. *Medical journal of Australia* 182(1): 17–19.

Kurniawati, H.; and Yadav, V. 2013. An Online POMDP Solver for Uncertainty Planning in Dynamic Environment. In *Proc. Int. Symp. on Robotics Research*.

Lantzch, R.; Greiser, S.; Wolfram, J.; Wartmann, J.; Müllhäuser, M.; Lüken, T.; Döhler, H.-U.; and Peinecke, N. 2012. ALLFLIGHT: Helicopter pilot assistance in all phases of flight. .

Le Blaye, P. 2003. A concept of Flight Execution Monitor (FEM) for helicopter pilot assistance. Technical report, OFFICE NATIONAL D'ETUDES ET DE RECHERCHES AERONOSPATIALES CEDEX FRANCE.

Lüken, T.; and Korn, B. 2007. PAVE: A prototype of a helicopter pilot assistant system. Technical report.

Mausam; and Kolobov, A. 2012. *Planning with Markov Decision Processes*. Morgan&Claypool.

Trevizan, F.; Teichteil-Königsbuch, F.; and Thiébaux, S. 2017. Efficient Solutions for Stochastic Shortest Path Problems with Dead Ends. In *Proc. of 33rd Int. Conf. on Uncertainty in Artificial Intelligence (UAI)*.

Trevizan, F.; Thiébaux, S.; and Haslum, P. 2017. Occupation Measure Heuristics for Probabilistic Planning. In *Proc. of 27th Int. Conf. on Automated Planning and Scheduling (ICAPS)*.

Trevizan, F.; and Veloso, M. 2012. Trajectory-Based Short-Sighted Probabilistic Planning. In *Advances in Neural Information Processing Systems (NIPS)*.