Safe path planning for UAV urban operation under GNSS signal occlusion risk

Jean-Alexis Delamer, Yoko Watanabe, Caroline Ponzoni Carvalho Chanel

Safe path planning for UAV urban operation under GNSS signal occlusion risk

Jean-Alexis Delamer [a], Yoko Watanabe [b,*], Caroline P.C. Chanel [c]

[a] School of Computing, Queen's University, Kingston, Canada
[b] ONERA - The French Aerospace Laboratory, Toulouse, France
[c] ISAE-SUPAERO, Université de Toulouse, Toulouse, France

Keywords:
Navigation
Path planning
POMDP
PO-SSP
UAV
Safety

A B S T R A C T

This paper introduces a concept of *safe path planning* for UAV's autonomous operation in an urban environment where GNSS-positioning may become unreliable or even unavailable. If the operation environment is a priori known and geo-localized, it is possible to predict a GNSS satellite constellation and hence to anticipate its signal occlusions at a given point and time. Motivated from this, our main idea is to utilize such sensor availability map in path planning task for ensuring UAV navigation safety. The proposed concept is implemented by a Partially Observable Markov Decision Process (POMDP) model. It incorporates a low-level navigation and guidance module for propagating the UAV state uncertainty in function of the probabilistic sensor availability. A new definition of cost function is introduced in this model such that the resulting optimal policy respects a user-defined safety requirement. A goal-oriented version of Monte-Carlo Tree Search algorithm, called POMCP-GO, is proposed for POMDP solving. The developed safe path planner is evaluated on two simple obstacle benchmark maps as well as on a real elevation map of San Diego downtown, along with GPS availability maps.

1. Introduction

1.1. UAV operation in urban environment

In recent years, Unmanned Aerial Vehicles (UAVs) or drones have started to be widely used in real-life operations such as package delivery, infrastructure inspection, disaster relief and rescue operation, and so on. Though, UAV operation in an urban or peri-urban environment is yet quite a challenge due to its immature level of navigation autonomy and safety. Most of outdoor UAVs rely its localization precision on GNSS (Global Navigation Satellite System). However, in an urban environment surrounded by tall buildings, GNSS is at risk of losing a line-of-sight to one or more satellites, and even worse, of receiving a signal via reflected path. Such signal occlusion and multipath effect make a GNSS position solution unreliable or even unavailable. If a drone does not have any alternative navigation sensor, such GNSS-failing situation could critically degrade its localization and trajectory execution accuracy, and lead a fatal collision or crash.

A quality of the GNSS position solution can be given as a metric called Position Dilution of Precision (PDOP), derived purely from a geometry of satellite constellation and the environment. Roughly speaking, it signifies a theoretical standard deviation of the positioning error. If the operation environment and geo-location are known a priori, it is possible to predict the PDOP value at a given point and time, and hence to build a PDOP map [1]. Fig. 1 shows an example of GPS(Global Positioning System)-PDOP map computed by loading an elevation map of San Diego downtown, available in [2], in the Oktal-SE GPS simulator.[1] In this example on the right figure, the GPS position solution has a risk of including large error in red areas (with large PDOP values), and hence the UAV navigation system should better not rely on GPS when traveling in them.

In order to cope with such possible degradation in navigation accuracy and to avoid fatal collision or crash, this paper addresses the problem of *safe path planning* for UAV urban operation. Motivated from the GNSS-PDOP map generation, a main idea is to make use of such environment-dependent sensor quality/availability information in path planning task for ensuring UAV flight safety with regard to collision risk. Before introducing the proposed approach, some related works are reviewed in the remaining of this section.

* Corresponding author.
E-mail addresses: j.delamer@queensu.ca (J.-A. Delamer), Yoko.Watanabe@onera.fr (Y. Watanabe), caroline.chanel@isae-supaero.fr (C.P.C. Chanel).

https://doi.org/10.1016/j.robot.2021.103800

[1] All of the GPS-PDOP maps used in this work were generated with this simulator available at Department of Electro Magnetism and Radar, ONERA.

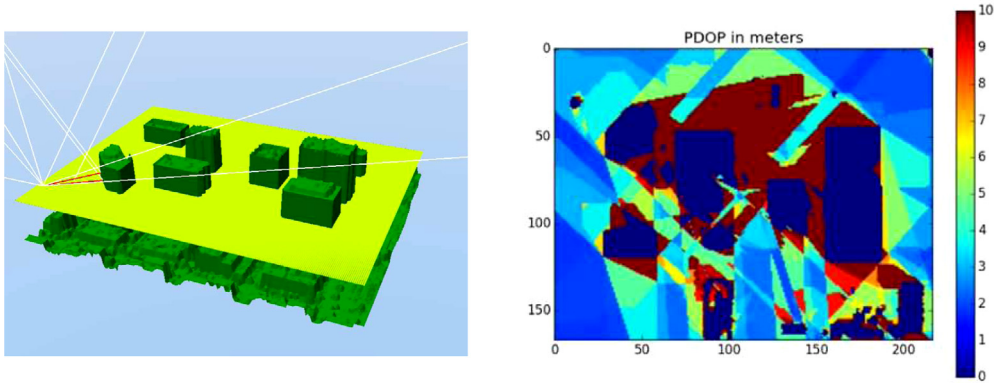


Fig. 1. Example of GPS-PDOP map generation: (Left) Digital elevation map of San Diego downtown is loaded in the Oktal-SE GPS simulator, which simulates GPS signal reception at a given location and time. Multipath effects are shown in red. (Right) Resulting GPS-PDOP map at a ground level. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

1.2. Deterministic path planning under motion or localization uncertainty

Given the UAV operation environment in a form of 3D model or elevation map, a classical flight path planning task consists of finding a path from a start point *A* to a destination *B* with a maximum *Efficiency* (i.e., minimum distance, energy or time) while avoiding obstacles. In the most classical approaches, localization and motion uncertainties are considered only globally by inflating obstacles for a certain fixed safety margin.

Refs. [3,4] propose Chance-Constrained path planning (CCP) approaches, based on the Mixed Integer Linear Programming. Given a time-invariant linear system with Gaussian noise representing motion uncertainty (disturbance), the CCP tries to find a cost-optimal path while explicitly limiting an obstacle collision probability to a user-defined threshold. The collision probability is evaluated with the Gaussian state distribution which is deterministically and environment-independently propagated along a planned path.

Refs. [5–9] implement a state estimation module in path planner for taking into account an evolution of the localization uncertainty along a planned path, and apply graph or tree search algorithms (such as Dijkstra, A*, RRT* [10], BRM [11] and RRBT [12]). Upon each node transition, the localization uncertainty is propagated by Kalman filter (KF) with a given local sensor precision or availability. For example, Ref. [5] uses a laser range-finder perception model with a limited sensing range, and the work presented in [6] uses a landmark map for visual SLAM (Simultaneous Localization and Mapping) navigation, both in a GNSS-denied environment. Refs. [7], as far as the authors know, is one of the first work proposing to consider a GNSS availability map in UAV path planning, where the propagated localization uncertainty is used not only in evaluating collision condition but also in cost function to find the most informative path. Recent work [8,13] also addresses a problem of UAV path planning by taking into account GNSS local precision and availability in an urban environment.

A drawback of these approaches is that graph node (or state-mean) transitions are deterministic. They make an assumption of so-called *Maximum Likelihood Observation* [14] where stochasticity in the sensor measurement is ignored. Although this assumption is commonly used to simplify the problem, large errors in navigation sensor measurements could drive the UAV state mean far from a deterministically-chosen destination node, and so invalidate the considered node transition.

1.3. Probabilistic path planning under motion and sensing uncertainties

Partially Observable Markov Decision Process (POMDP) framework offers a general mathematical formulation for sequential decision-making problems under both motion and sensing uncertainties [15,16]. One can find works that have applied discrete/continuous POMDPs to solve robotic task, motion or path planning under uncertainty.

1.3.1. Discrete POMDP path planning

Several works have applied discrete POMDPs (i.e. state, action and observation are discrete variables) to solve robotic task or path planning under uncertainty [17–22]. Those robotic POMDP applications usually approach a policy solution thanks to heuristic guided algorithms that explore only reachable belief states for computing a value function expressing the expected total cost of the policy being applied.

Ref. [21] addresses an Autonomous Underwater Vehicle (AUV) navigation problem under localization and motion uncertainty and possible obstacle collision risk. This work actually proposes and applies an extension of the POMDP framework, called Mixed-Observability Markov Decision Process (MOMDP). As explained later in this paper, MOMDP factorizes the state space into fully and partially observable state variables, which enables to decrease the belief state dimension resulting in less computational effort needed to approach the value function and the associated policy [23]. Although [21] proves the competitive policy computation time of MOMDP planning compared to classical POMDP model, their discrete POMDP model for the AUV navigation problem remains small. The policy computation will be more challenging when the state, action and/or observation spaces are continuous, which is the case for the motion or trajectory planning context.

1.3.2. Continuous POMDP path planning

Refs. [24,25] implement LQG (Linear Quadratic Gaussian) feedback controller along with the vehicle dynamics into POMDP model for taking into account its closed-loop motion uncertainty. These continuous POMDP path planners consider the sensor measurement as an observation and the measurement noise as sensing uncertainty, with an assumption that the environment-dependent sensing quality is perfectly known.

Unlike those works, this paper introduces uncertainty in the a priori knowledge on the local sensor quality/availability map. As mentioned earlier, the GNSS-PDOP map can be generated from the geo-referenced 3D environment model and the GNSS satellite

constellation prediction at a given time. Hence, uncertainties in those used models and parameters will induce uncertainty in the resulting PDOP. Indeed, the experimental results presented in [8] emphasizes this point. They compared a predicted GPS mean position error with that obtained by a real GPS receiver for different environment context such as open-sky and narrow urban canyon. Certain results show an important gap, which confirms our interest of considering an uncertainty in the predicted sensor quality/availability map.

1.4. Proposed safe path planning approach

This paper firstly defines a MOMDP model for our UAV safe path planning problem, where the flight time is minimized while respecting a user-defined maximum allowable collision risk under the UAV state uncertainty that may evolve in function of the motion execution error and the probabilistic local sensor availability. The proposed MOMDP model is quite different from the ones seen in most of the related work, and has the following particularities for coping with the new and challenging features addressed in the problem:

- It incorporates a low-level navigation and guidance module to propagate the UAV state uncertainty in continuous state space, **without making the assumption of *Maximum Likelihood Observation***.
- The observation is defined as sensor availability but not as sensor measurement, which is not accessible at the moment of planning. It enables to avoid working with a continuous observation variable. **Sensor availability is also considered as an uncertain fully observable state** and given by a probability grid map. These specific assumptions make a **belief state** non-Gaussian.
- Collision penalty is not constant, but depends on the time-to-collision. This definition makes the resulting MOMDP value function linear to a total collision probability, which allows us to choose the collision penalty value according to **a user-defined maximum allowable collision risk**.

Moreover, this paper presents a new algorithm, called POMCP-GO, for solving this proposed MOMDP safe path planning problem. Inspired by the RTDP-bel algorithm [26], POMCP-GO extends POMCP (POMDP Monte-Carlo Planning) [27] to a Goal-Oriented version of the Monte-Carlo Tree Search algorithm applied for partially observable domains.

The remainder of this paper is organized as follows: Section 2 provides a MOMDP model of our safe path planning problem. Section 3 introduces a method to define the collision penalty cost in function of the allowable collision risk. Section 4 presents the proposed POMCP-GO algorithm. Simulation results are then presented in Section 5, followed by conclusion and future work.

2. MOMDP-based model for UAV safe path planning

The safe path planning problem addressed in this paper consists of finding safe (avoiding obstacles) and efficient (minimum time or distance) trajectories towards a goal under uncertainty. This sequential decision making problem can be defined as a Partially Observable Stochastic Shortest Path (PO-SSP) planning problem [28], which is an extension of Stochastic Shortest Path (SSP) problem [29,30] to the case of imperfect state information. This section first describes our planning objective and hypotheses. Then, formulating our safe path planning problem as PO-SSP, a MOMDP model is proposed.

2.1. Planning objective and hypotheses

This paper considers a problem of finding a path for an UAV to reach a given goal region in an urban environment, whose model is given as a 3D grid occupancy map. The UAV is equipped with a classical navigation sensor setup, IMU (Inertial Measurement Unit) and GNSS, for its localization. The IMU measurement is always available regardless the vehicle position in the environment, but it includes an unknown bias which needs to be compensated by using other unbiased sensor measurement notably GNSS position.

Availability of the GNSS measurement depends on the environment and its probability is given in the same 3D grid map as the occupancy map. In this work, a probability of GNSS availability is computed for each grid cell by setting a maximum position error threshold to a zero-mean Gaussian distribution with a standard deviation given by a simulated PDOP value (Fig. 1). Note that, the PDOP value slowly changes over time due to the movement of the GNSS satellite constellation. However, this work suppose that the mission duration is short enough to assume the PDOP map as static (up to 5-min missions for 20-min validity of the PDOP map).

For simplicity, it is supposed that the UAV estimates its attitude with accuracy from IMU and other sensors, and that the low-level controller realizes an acceleration command with negligible delay. Hence, only the vehicle translation kinematics model is considered for its motion dynamics.

Due to the uncertainties in motion, sensing and sensing availability, an evolution of the UAV state as well as that of the state distribution become stochastic. Under such uncertainties, no navigation strategy can bring the UAV to reach in the goal region nor avoid collision with probability 1. Therefore, our planning objective is to find a policy to minimize an expected total path cost (i.e. flight time) to the goal while ensuring flight safety by a collision probability not exceeding a user-defined threshold. Seen it as a Risk-Constrained Stochastic Shortest Path problem with unavoidable deadends (RC-SSPUDE) [31] with the partial observable state, this paper proposes to adopt an MOMDP planning model [21,23]. Instead of handling the risk constraint explicitly in the optimization (as done in the CCP approaches [3,4]), here introduced a particular cost function which enables to penalize both flight time and collision in a way that makes the resulting optimal policy respect the maximum collision risk.

Fig. 2 gives an overview of the assumed architecture of the proposed MOMDP model. It incorporates a low-level navigation and guidance system along with the vehicle kinematics (called GNC module) to propagate the probability density (i.e. belief state) of the UAV state vector. Assuming that the UAV always knows whether GNSS position measurement is accurate enough to be used in the navigation or not at the current decision time step, the GNSS availability is included as an observable state variable of the MOMDP model that updates the belief state. In this sense, the planning module does not have a direct access to the UAV state (e.g. position, velocity) but only to its probability distribution.

The next subsections will present the GNC module and the proposed MOMDP model.

2.2. GNC module

Fig. 3 illustrates a detail of the GNC module used in the proposed MOMDP planning model. It is composed of the UAV motion model, a guidance law, a state estimator (navigation) and the IMU and GNSS sensor models. Similar to the work presented in [32], a closed-loop of these component will propagate the UAV state, for an action execution, in a stochastic manner.

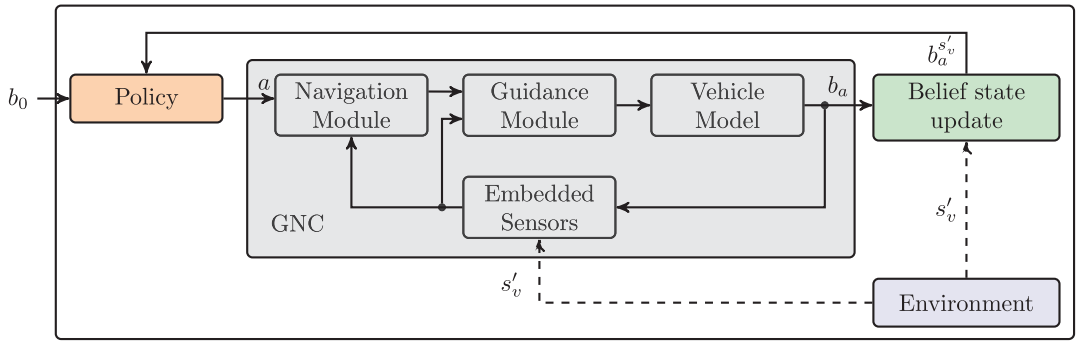


Fig. 2. The MOMDP safe path planning architecture. The GNC module is incorporated as a part of the planning model.

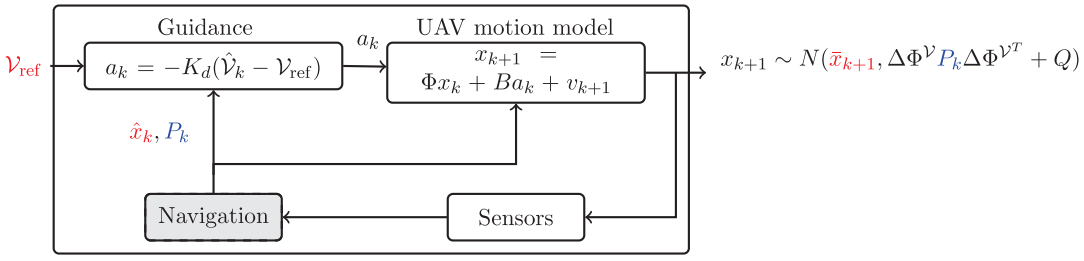


Fig. 3. Detailed GNC module. It is a closed-loop of the vehicle motion model, a guidance law, and a Kalman filter for the UAV state estimation (navigation) with the IMU/GNSS sensor models.

2.2.1. UAV state and motion model

As stated in Section 2.1, the UAV motion is represented only by its translational kinematics. Let $\boldsymbol{X}$, $\boldsymbol{V}$ and $\boldsymbol{a}$ be the UAV's position, velocity and acceleration vectors, respectively, in a reference frame fixed to the environment. The non-gravitational acceleration is measured by IMU 3-axis accelerometers in the UAV-fixed frame. But this measurement includes an unknown bias $\boldsymbol{b}_a$ which needs to be estimated and compensated. Hence, we define the UAV state vector $\boldsymbol{x}$ by its position, velocity and accelerometer bias:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{X}^T & \boldsymbol{V}^T & \boldsymbol{b}_a^T \end{bmatrix}^T \tag{1}$$

Then its discretized motion model from a time step t_k to $t_{k+1} = t_k + \Delta t$ is given by

$$\begin{aligned} \boldsymbol{x}_{k+1} = \begin{bmatrix} \boldsymbol{X}_{k+1} \\ \boldsymbol{V}_{k+1} \\ \boldsymbol{b}_{ak+1} \end{bmatrix} &= \begin{bmatrix} I & (\Delta t)I & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_k \\ \boldsymbol{V}_k \\ \boldsymbol{b}_{ak} \end{bmatrix} \\ &\quad + \begin{bmatrix} (\frac{1}{2}\Delta t^2)I \\ (\Delta t)I \\ 0 \end{bmatrix} \boldsymbol{a}_k + \begin{bmatrix} \boldsymbol{v}_{X\,k+1} \\ \boldsymbol{v}_{V\,k+1} \\ \boldsymbol{v}_{b_a\,k+1} \end{bmatrix} \\ &= \Phi \boldsymbol{x}_k + B\boldsymbol{a}_k + \boldsymbol{v}_{k+1} \end{aligned} \tag{2}$$

where $\boldsymbol{v}_{k+1}$ is a zero-mean Gaussian noise with covariance Q, denoted as $\boldsymbol{v}_{k+1} \sim \mathcal{N}(\boldsymbol{0}, Q)$.

2.2.2. Navigation system for the UAV state estimation

A Kalman filter (KF) is applied to estimate the UAV state vector $\boldsymbol{x}_k$ from the accelerometer measurement and the GNSS position and velocity measurements, if available. Let $\hat{\boldsymbol{x}}_k$, $\tilde{\boldsymbol{x}}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k$ and $P_k = \mathbb{E}\left[\tilde{\boldsymbol{x}}_k \tilde{\boldsymbol{x}}_k^T\right]$ be the state estimate, its estimation error and error covariance matrix at the time step t_k. Firstly, a prediction step is performed based on the motion model (2) and the IMU acceleration measurement, modeled by

$$\boldsymbol{a}_{IMU\,k} = R^T(\boldsymbol{q}_k)(\boldsymbol{a}_k - \boldsymbol{g}) + \boldsymbol{b}_{ak} + \boldsymbol{\xi}_{ak} \tag{3}$$

where $\boldsymbol{q}_k$ is a known UAV attitude and $R(\boldsymbol{q}_k)$ represents a rotation matrix from the UAV frame to the reference frame. $\boldsymbol{g}$ is a gravity vector in the reference frame, and $\boldsymbol{\xi}_{ak} \sim \mathcal{N}(\boldsymbol{0}, R_a = \sigma_a^2 I)$ is a measurement noise. By substituting (3) into (2),

$$\begin{aligned} \boldsymbol{x}_{k+1} &= (\Phi - BR(\boldsymbol{q}_k)\begin{bmatrix} O & O & I \end{bmatrix})\boldsymbol{x}_k + B(R(\boldsymbol{q}_k)\boldsymbol{a}_{IMU\,k} + \boldsymbol{g}) \\ &\quad + \boldsymbol{v}_{k+1} - BR(\boldsymbol{q}_k)\boldsymbol{\xi}_{ak} \\ &= \Phi_{ak}\boldsymbol{x}_k + B\bar{\boldsymbol{a}}_{IMU_k} + \bar{\boldsymbol{v}}_{k+1} \end{aligned} \tag{4}$$

where the augmented noise follows $\bar{\boldsymbol{v}}_{k+1} \sim \mathcal{N}(\boldsymbol{0}, \bar{Q}_{k+1} = Q + BR_aB^T)$. By using (4), the KF prediction step is performed as follows.

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1}^- &= \Phi_{ak}\hat{\boldsymbol{x}}_k + B\bar{\boldsymbol{a}}_{IMU_k} \\ P_{k+1}^- &= \mathbb{E}\left[\tilde{\boldsymbol{x}}_{k+1}^- \tilde{\boldsymbol{x}}_{k+1}^{-T}\right] = \Phi_{ak} P_k \Phi_{ak}^T + \bar{Q}_{k+1} \end{aligned} \tag{5}$$

It should be noted that, although an acceleration command $\boldsymbol{a}_k$ is assumed in the vehicle motion model (2) for simplicity, it is not accessible in the real navigation system and hence cannot be directly used in the KF prediction.

The GNSS measures the UAV position and velocity in the reference frame:

$$\boldsymbol{z}_{k+1} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \end{bmatrix} \boldsymbol{x}_{k+1} + \begin{bmatrix} \boldsymbol{\xi}_{X\,k+1} \\ \boldsymbol{\xi}_{V\,k+1} \end{bmatrix} = H\boldsymbol{x}_{k+1} + \boldsymbol{\xi}_{k+1} \tag{6}$$

where $\boldsymbol{\xi}_{k+1} \sim \mathcal{N}(\boldsymbol{0}, R_{k+1})$. As seen in the PDOP map (Fig. 1), the GNSS measurement quality depends on the environment and could significantly degrade due to multi-path effect. However, if the receiver does not aware of that, it will provide the erroneous solution associated with a small error variance, which should not be trusted. For this reason, regardless the PDOP value, a constant error covariance matrix $R_{k+1} = R_{GNSS}$ is used in the navigation filter design. For example, we can use the same value as the position accuracy threshold used in the GNSS availability probability map generation (explained in Section 2.1). Based on this sensor model (6), when the GNSS measurement is available, the KF correction is applied to the state prediction (5) as below.

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1} &= \hat{\boldsymbol{x}}_{k+1}^- + K_{k+1}(\boldsymbol{z}_{k+1} - H\hat{\boldsymbol{x}}_{k+1}^-) \\ P_{k+1} &= \mathbb{E}\left[\tilde{\boldsymbol{x}}_{k+1} \tilde{\boldsymbol{x}}_{k+1}^T\right] = (I - K_{k+1}H)P_{k+1}^- \end{aligned} \tag{7}$$

where K_{k+1} is a Kalman gain derived by $K_{k+1} = P^-_{k+1}H^T(HP^-_{k+1}H^T + R_{GNSS})^{-1}$. When the GNSS measurement is not available, $\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}^-_{k+1}$ and $P_{k+1} = P^-_{k+1}$.

2.2.3. Guidance law for velocity tracking

A linear feed-back guidance law for velocity tracking is applied to guide the UAV to a desired direction, determined by a selected action defined in the planning model. Let $\boldsymbol{V}_{ref}$ is a desired velocity. Then the following acceleration input is applied.

$$\boldsymbol{a}_k = -K_d(\hat{\boldsymbol{V}}_k - \boldsymbol{V}_{ref}) \tag{8}$$

where $K_d > 0$ is a control gain. By substituting this into (2),

$$\boldsymbol{x}_{k+1} = (\Phi - \Delta\Phi_V)\boldsymbol{x}_k + BK_d\boldsymbol{V}_{ref} + \Delta\Phi_V\tilde{\boldsymbol{x}}_k + \boldsymbol{v}_{k+1} \tag{9}$$

where $\Delta\Phi_V = BK_d\begin{bmatrix}O & I & O\end{bmatrix}$. Given the UAV state $\boldsymbol{x}_k$, its estimation error covariance P_k and the desired velocity $\boldsymbol{V}_{ref}$, the state distribution at t_{k+1} becomes

$$\begin{aligned}\boldsymbol{x}_{k+1} \sim \mathcal{N}(\boldsymbol{\mu}_{k+1} &= (\Phi - \Delta\Phi_V)\boldsymbol{x}_k + BK_d\boldsymbol{V}_{ref},\ \Sigma_{k+1} \\ &= \Delta\Phi_V P_k \Delta\Phi_V^T + Q)\end{aligned} \tag{10}$$

After N time steps from t_k, the state distribution becomes $\boldsymbol{x}_{k+N} \sim \mathcal{N}(\boldsymbol{\mu}_{k+N}, \Sigma_{k+N})$ with the mean and the covariance matrix are incrementally derived as follows.

$$\begin{aligned}\boldsymbol{\mu}_{k+N} &= (\Phi - \Delta\Phi_V)\boldsymbol{\mu}_{k+N-1} \\ \Sigma_{k+N} &= (\Phi - \Delta\Phi_V)\Sigma_{k+N-1}(\Phi - \Delta\Phi_V)^T + \Delta\Phi_V P_{k+N-1}\Delta\Phi_V^T + Q\end{aligned} \tag{11}$$

where the estimation error covariance P is propagated at the same time by the Kalman filter. This closed-loop path execution error distribution will be implemented as a part of the state transition function in the MOMDP model.

2.3. MOMDP safe path planning model

Ong et al. [21], followed by Araya et al. [23] have proposed a special class of the classical POMDP framework, called Mixed-Observability Markov Decision Process (MOMDP). This model factorizes the state space into fully and partially observable state variables, which reduces the belief state space dimension and thus policy computation time. Since our safe path planning problem considers the partially observable UAV state vector $\boldsymbol{x}$ and the fully observable sensor availability, it can adopt this MOMDP model.

The MOMDP is here defined as a tuple $(\mathcal{S}_v, \mathcal{S}_h, \mathcal{A}, \Omega, \mathcal{T}_v, \mathcal{T}_h, \mathcal{O}, \mathcal{C}, \mathcal{G}, b_0)$, where $\mathcal{S}_v$ is the fully observable state space, $\mathcal{S}_h$ is hidden (partially observable) state space. Their combination constitutes the state space of the POMDP such as $\mathcal{S} = \mathcal{S}_v \times \mathcal{S}_h$. Since $\mathcal{S}_v$ is visible, the belief state space can be partitioned as $\mathcal{B} = \mathcal{S}_v \times \mathcal{B}_h$. In a same fashion, the state transition function is factorized into $\mathcal{T}_v$ for the fully observable state variables, and $\mathcal{T}_h$ for the hidden state variables. Ω and $\mathcal{O}$ are the observation space and function. $\mathcal{C}$ is the cost function. $\mathcal{G} \subset \mathcal{S}$ defines a set of goal states, and $b_0 = (s_{v_0}, b_{h_0})$ is the initial belief.

The Fig. 4a schematizes the MOMDP planning model proposed in this paper. It actually differs from the original MOMDP [21,23]. In our case, the fully observable state s_v depends on the hidden state s_h, but not the contrary. It will be shown that this assumption implies particular transition and observation functions. In what follows, the detailed definitions of each element of the proposed MOMDP safe path planning model are presented.

2.3.1. States, goal states, actions and observations

The fully observable state $s_v \in \mathcal{S}_v$ is defined as a tuple $s_v = (F_{GNSS}, F_C, P, \Theta)$, where $F_{GNSS}, F_C \in \{0, 1\}$ are Boolean flags to indicate the GNSS sensor availability and the collision to obstacles, respectively. $P \in S^9_{++}(\mathbb{R})$ is the UAV localization error covariance matrix[2] computed by the Kalman filter (5)–(7). It should be noted that, despite the stochasticity in the sensor measurements and in the guidance command, the estimation error covariance matrix P can be calculated in a deterministic way. $\Theta \in \mathbb{R}^+_0$ represents a total flight time from the initial state s_0 until s. Since this work assumes a known execution time for all actions, an evolution of Θ is also deterministic. For their deterministic transition from known initial state, P and Θ can be considered as a fully observable state variable. The hidden state $s_h \in \mathcal{S}_h$ coincides with the UAV state vector defined in (1), $s_h = \boldsymbol{x} \in \mathbb{R}^9$. The Fig. 4b illustrates state variable structure and their dependencies.

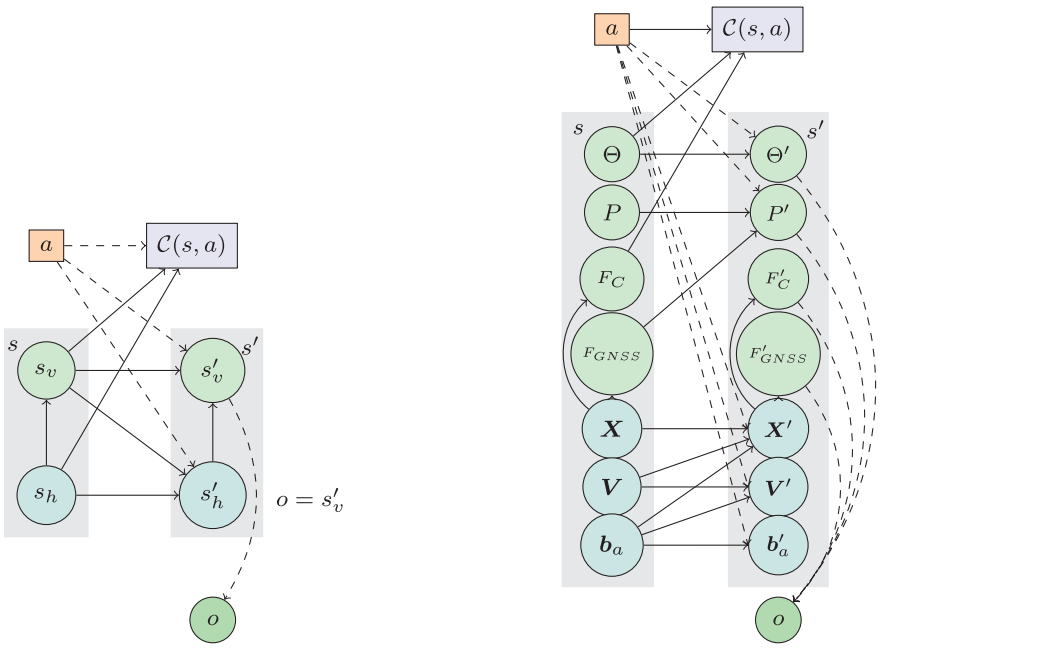
The goal states $\mathcal{G} \subset \mathcal{S}$ are defined as the set of all the states in which the position is included in a bounded region $\mathcal{G}_X \subset \mathbb{R}^3$ around a given goal position $\boldsymbol{X}_g$. Let us define the goal hidden state space by $\mathcal{G}_h = \{s_h = \boldsymbol{x} = [\boldsymbol{X}, \boldsymbol{V}, \boldsymbol{b}_a] \in \mathcal{S}_h | \boldsymbol{X} \in \mathcal{G}_X\}$. Then, the goal state space is given by $\mathcal{G} = \{s = (s_v, s_h) \in \mathcal{S} | s_h \in \mathcal{G}_h\}$. Goal states are considered as absorbing states. In addition, an artificial terminal state s_T is added to the state space for easing further developments. Let us define a collision visible state space by $\mathcal{S}_{v_C} = \{s_v = (F_{GNSS}, F_C, P, \Theta) \in \mathcal{S}_v | F_C = 1\}$, and a collision state space by $\mathcal{S}_C = \{s = (s_v, s_h) \in \mathcal{S} | s_v \in \mathcal{S}_{v_C}\}$. The terminal state s_T is an absorbing state, mandatory reached from any collision state $s \in \mathcal{S}_C$. Thus, the set of absorbing states can be defined as $\mathcal{S}_T = \mathcal{G} \cup \{s_T\}$.

An action $a \in \mathcal{A}$ corresponds to a selection of the desired velocity $\boldsymbol{V}_{ref} \in \mathbb{R}^3$ to be given to the guidance law in (8). For simplification, this paper considers a discrete action space by limiting it to a finite set. For instance, it could be $\{\boldsymbol{V}_{North}, \boldsymbol{V}_{South}, \boldsymbol{V}_{East}, \boldsymbol{V}_{West}, \boldsymbol{V}_{Up}, \boldsymbol{V}_{Down}\}$. It is assumed that a value of $\boldsymbol{V}_{ref}$ holds for a ΔT_a period of time. Note, it has a subscription a to indicate a specific action since this action duration can differ from one action to other.

A specificity of the MOMDP model proposed in this paper is that the sensor measurements are not considered as observations. Such sensor measurements, defined in a continuous space, are assumed not accessible at the planning level. In result, if we treat them as observation in the POMDP model, policy computation needs to consider a decision tree with infinitely many branches because observation strategies becomes a continuous function (i.e. all possible combinations of actions and observations in a given horizon $h = (b_0, a_0, o_1, a_1, o_2, \ldots, a_{t-1}, o_t)$) [33]. Many of the related work avoid this issue by making the aforementioned *Maximum Likelihood Observation* assumption [14] and ignoring the measurement residual. Unlike these, our MOMDP model keeps the stochasticity in the sensor measurements (as well as in the guidance input) and includes it in the state transition function.

In our model, the set of observations Ω is equal to $\mathcal{S}_v$ (see Fig. 4b) which is composed of the discrete state variables (F_{GNSS}, F_C) and the deterministic continuous state variables (P, Θ). This way of modeling limits the decision tree branching factor because observation strategies are finite resulting in a finite reachable belief state set (whereas possibly huge), over which the policy is defined. Although the agent receives no direct observation on the state s_h, the observation $o = s_v$ will modify its distribution thanks to the dependence of s_v on s_h (Fig. 4a). In consequence, the complete state $s \in \mathcal{S}$ remains partially observable.

[2] $S^n_{++}(\mathbb{R})$ denotes a set of positive definite matrices with a dimension n.



(a) The modified MOMDP model. (b) The MOMDP state variables structure and dependencies.

Fig. 4. The modified MOMDP planning model. Note the fully observable state s_v depends on the hidden state s_h, and the observation o is assumed to be equal to s_v. Despite of this assumption the complete state s remains partially observable.

2.3.2. Transition and observation functions

The state transition function defines a state distribution after executing an action $a \in \mathcal{A}$ at the state $s \in \mathcal{S}$. The execution duration ΔT_a of each action a corresponds to a decision step of the planner, which is in general much longer than a time discretization step Δt (introduced in (2)) of the lower-level GNC module. This paper defines one decision step by several time steps of the GNC module, i.e., $\Delta T_a = N_a \Delta t$ with $N_a > 1$ for an action a.

The state transition function is factorized as $\mathcal{T}(s_v, s_h, a, s'_v, s'_h) = \mathcal{T}_v(s_v, a, s'_v, s'_h)\mathcal{T}_h(s_v, s_h, a, s'_h)$ where :

- $\mathcal{T}_h(s_v, s_h, a, s'_h) = \Pr(s'_h|s_v, s_h, a) \sim \mathcal{N}(\bar{s}'_h, \sum')$, which is based on the GNC closed-loop vehicle motion model (11) with the number of time steps $N = N_a$.
- $\mathcal{T}_v(s_v, a, s'_v, s'_h) = \Pr(s'_v|s_v, a, s'_h)$, representing the transition function for s'_v, whose stochasticity basically comes from the probability of the GNSS availability F_{GNSS} which depends on the hidden UAV position. Transition of the three other state variables in s'_v, given s'_h, is deterministic. That is, for a probability 1, the collision flag becomes $F'_C = 1$ when the position in s'_h falls in an occupied grid of the environment map (otherwise $F'_C = 0$), the localization error covariance matrix P' is obtained after iterating the Kalman filter process (5), (7) for N_a time steps from $(s_h = \boldsymbol{x}, P)$, and the flight time is updated by $\Theta' = \Theta + \Delta T_a$.

However, the transition function is defined differently for the following two particular cases:

1. a free-cost transition to the terminal state s_T is imposed from any collision state, i.e., $\mathcal{T}(s_v, s_h, a, s' = s_T) = 1$, for $\forall s_v \in \mathcal{S}_{v_C}$.
2. the definition of absorbing states holds for the goal and the terminal state, thus, $\mathcal{T}(s, a, s' = s) = 1$ for $\forall s \in \mathcal{S}_T = \mathcal{G} \cup \{s_T\}$.

Since our observation $o \in \Omega$ coincides with the fully observable state $s'_v \in \mathcal{S}_v$, the observation function is defined as $\mathcal{O}(o, s'_v) = \Pr(o|s'_v) = \delta(o, s'_v)$[3] meaning that $\Pr(o = s'_v) = 1$.

2.3.3. Belief state update

After the state transition from s to s' via the action a, the agent perceives an observation $o \in \Omega$ following the observation function $\mathcal{O}$. Given the previous belief state $b = (s_v, b_h) \in \mathcal{S}_v \times \mathcal{B}_h$ and the transition and observation functions, one can apply the Bayes' rule to update the state distribution. The belief update step of the proposed model, developed below, differs from previous works as the visible state depends on the hidden state.

$$
\begin{aligned}
b^o_a(s'_v, s'_h) = \Pr(s'_v, s'_h|s_v, b_h, a, o) &= \frac{\Pr(s'_v, s'_h, o|s_v, b_h, a)}{\Pr(o|s_v, b_h, a)} \\
&= \frac{\Pr(o|s'_v)\Pr(s'_v|s_v, b_h, a, s'_h)\Pr(s'_h|s_v, b_h, a)}{\sum_{s'_v \in \mathcal{S}_v} \delta(o, s'_v) \int_{s'_h \in \mathcal{S}_h} \Pr(s'_v, s'_h|s_v, b_h, a) ds'_h} \\
&= \frac{\delta(o, s'_v)\Pr(s'_v|s_v, a, s'_h) \int_{s_h \in \mathcal{S}_h} \Pr(s'_h|s_v, s_h, a) b_h(s_h) ds_h}{\sum_{s'_v \in \mathcal{S}_v} \delta(o, s'_v) \int_{s'_h \in \mathcal{S}_h} \Pr(s'_v|s_v, a, s'_h) \int_{s_h \in \mathcal{S}_h} \Pr(s'_h|s_v, s_h, a) b_h(s_h) ds_h ds'_h} \\
&= \delta(o, s'_v) \cdot \frac{\Pr(s'_v|s_v, a, s'_h) \int_{s_h \in \mathcal{S}_h} \Pr(s'_h|s_v, s_h, a) b_h(s_h) ds_h}{\int_{s'_h \in \mathcal{S}_h} \Pr(s'_v = o|s_v, a, s'_h) \int_{s_h \in \mathcal{S}_h} \Pr(s'_h|s_v, s_h, a) b_h(s_h) ds_h ds'_h}
\end{aligned}
$$

The last line of the above equation used a fact that $\delta(o, s'_v) = 0$ for $\forall s'_v \neq o$ to remove the summation in the denominator. Thus, in the following of this work the belief state update rule will refer directly to the distribution of the hidden state s'_h given s'_v:

$$
\begin{aligned}
b^{s'_v}_a(s'_h) &= \Pr(s'_h|s_v, b_h, a, s'_v) \\
&= \frac{\Pr(s'_v|s_v, a, s'_h) \int_{s_h \in \mathcal{S}_h} \Pr(s'_h|s_v, s_h, a) b_h(s_h) ds_h}{\int_{s'_h \in \mathcal{S}_h} \Pr(s'_v|s_v, a, s'_h) \int_{s_h \in \mathcal{S}_h} \Pr(s'_h|s_v, s_h, a) b_h(s_h) ds_h ds'_h} \\
&= \frac{\mathcal{T}_v(s_v, a, s'_v, s'_h) \int_{s_h \in \mathcal{S}_h} \mathcal{T}_h(s_v, s_h, a, s'_h) b_h(s_h) ds_h}{\int_{s'_h \in \mathcal{S}_h} \mathcal{T}_v(s_v, a, s'_v, s'_h) \int_{s_h \in \mathcal{S}_h} \mathcal{T}_h(s_v, s_h, a, s'_h) b_h(s_h) ds_h ds'_h}
\end{aligned} \tag{12}
$$

[3] A definition of the delta function used in this paper is given as $\delta(x, y) = 1$ if $x = y$, and 0 if $x \neq y$.

It is noteworthy that the updated belief state $b_a^{s'_v}(s'_h)$ becomes non-Gaussian even when the initial belief $b_h(s_h)$ is so. This is because the transition function $\mathcal{T}_v(s_v, a, s'_v, s'_h)$ used in (12) follows the GNSS availability probability given in a 3D grid map. Therefore, the computation of $\Pr(s'_v|s_v, b_h, a)$ becomes costly. For coping with it, this paper proposes to solve this path planning model by a sampling-based algorithm (Section 4).

2.3.4. Cost function

The cost function $\mathcal{C} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_0^+$ gives a cost of performing an action $a \in \mathcal{A}$ in the state $s \in \mathcal{S}$, and is defined as below in our model.

$$\mathcal{C}(s, a) = \mathcal{C}(s_v, s_h, a) = \begin{cases} 0 & \text{if } s \in \mathcal{S}_T = \mathcal{G} \cup \{s_T\} \\ K - \Theta & \text{if } s_v \in \mathcal{S}_{v_C} \\ \Delta T_a & \text{otherwise} \end{cases} \tag{13}$$

where ΔT_a is an action execution (i.e. flight) time and $K > 0$ is a fixed collision penalty cost. When a collision occurs, the cost of any action is given by this fixed penalty K subtracted with the total flight time Θ. This special condition avoids penalizing more the collision near the goal. In other words, the collision penalty has the same impact, wherever it occurs, on the value function at the initial belief. As explained later, this particular cost function makes the value function linear to the total collision probability when following a policy from the initial belief. By benefiting from this fact, this paper will propose in Section 3 a method to determine a value for the collision penalty K so that the resulting optimal policy will satisfy the user-defined maximum collision risk.

2.3.5. Value function and policy

The aim of solving a MOMDP problem is to find a policy $\pi : \mathcal{B} \to \mathcal{A}$, where $\mathcal{B}$ defines the belief state space (i.e. $\mathcal{B} = \mathcal{S}_v \times \mathcal{B}_h$), which optimizes a given criterion usually defined by a value function. In the PO-SSP planning problem, the value function $V^\pi(b)$ is defined as the expected total cost when starting from $b \in \mathcal{B}$ and following a policy π.

$$V^\pi(b) = \mathbb{E}\left[\sum_{t=0}^{\infty} \mathcal{C}(b_t, \pi(b_t))|\, b_0 = b\right] \tag{14}$$

where $\mathcal{C}(b_t, \pi(b_t) = a) = \mathbb{E}\left[\mathcal{C}(s, a)|b_t\right] = \sum_{s\in\mathcal{S}} \mathcal{C}(s, a)b_t(s)$ is the expected cost of executing an action $a \in \mathcal{A}$ in the belief state $b_t \in \mathcal{B}$. The optimal policy π^* is defined by a policy minimizing the value function, which results in the optimal value function $V^*(b) = V^{\pi^*}(b) = \min_\pi V^\pi(b)$.

Developing the summation in (14) and computing the expected value of the immediate future belief states based on their probabilities, it is straightforward to find that the optimal value function (for our MOMDP) can be expressed by the Bellman's equation as follows.

$$V^*(b) = \min_{a\in\mathcal{A}} Q(b, a) \tag{15}$$

$$\pi^*(b) = \arg\min_{a\in\mathcal{A}} Q(b, a) \tag{16}$$

where the Q-value of an action is defined as the value of performing an action a in belief state b, assuming the optimal policy will be followed afterwards:

$$Q(b, a) = \mathcal{C}(b, a) + \sum_{s'_v\in\mathcal{S}_v} \Pr(s'_v|b, a)V^*(s'_v, b_a^{s'_v}) \tag{17}$$

$b_a^{s'_v}$ is the hidden state distribution (12) given s'_v after executing an action a in the belief state b. The probability of having s'_v is given by the transition function, as done in the denominator of (12). One can then apply dynamic programming to compute (or to approximate) the optimal value function V^* and the related policy π^*. However, we recall here that the computation of $\Pr(s'_v|b, a)$ becomes costly for our particular MOMDP model. Thus, this paper proposes to approximate the Q-value of reachable belief states by a sampling-based algorithm, which will be presented in Section 4.

3. Collision penalty value for safety requirement

As stated in Section 2.1, our planning objective is to find a policy to minimize an expected total flight time to reach the goal while ensuring flight safety by not exceeding the maximum allowable collision probability threshold, say p_{thd}. Contrary to some related approaches [34,35] which consider explicitly such risk constraint in the planning model, this work treats this *Safety Requirement* implicitly by imposing the collision penalty K in (13). The idea is similar to what is done in the [36] where a fixed cost is assigned to dead-end states. Intuitively, the collision penalty cost specifies a point of compromise between the collision risk avoidance (*Safety*) and the flight time minimization (*Efficiency*). The larger the penalty is, the more priority is put on the *Safety*. Therefore, setting a right value to K is an important key to make the resulting optimal policy respect the safety requirement. This paper proposes a systematic way to determine an appropriate K value for a user-defined maximum collision risk p_{thd}.

3.1. Redefinition of value function

Firstly, this section shows that the value function with our cost function definition (13) becomes a linear combination of the expected flight time to the goal, and the expected collision probability with a coefficient equal to the penalty cost K.

Similarly to the collision state space, let us define a collision belief state space by $\mathcal{B}_C = \{b = (s_v, b_h) \in \mathcal{B}|s_v \in \mathcal{S}_{v_C}\}$. Recalling that every action brings the collision state to the absorbing terminal state s_T, we have $\Pr(s' = s_T|b \in \mathcal{B}_C, a) = 1$ for $\forall a \in \mathcal{A}$. From the definitions (14) with (13), for any policy π, the value function at any collision belief state $b_C \in \mathcal{B}_C$ becomes

$$V^\pi(b_C) = \mathcal{C}(b_C, \pi(b_C)) + \sum^{\infty} \mathcal{C}(s_T, \pi(s_T)) = K - \Theta \tag{18}$$

By using (18) and the fact that the goal states $s \in \mathcal{G}$ are absorbing, the value function at a non-collision belief state $b = (s_v, b_h) \notin \mathcal{B}_C$ can be expanded as follows.

$$\begin{aligned} V^\pi(b) &= p^\pi_{C_1}(b)(K - \Theta) \\ &\quad + \left(1 - p^\pi_{G_0}(b) - p^\pi_{C_1}(b)\right) \\ &\quad \times \left(\Delta T_{\pi(b)} + \mathbb{E}\left[V^\pi(b')\middle| b, \pi(b), s \notin \mathcal{G}, b' \notin \mathcal{B}_C\right]\right) \\ &= p^\pi_{C_1}(b)K + p^\pi_{G_0}(b)\Theta \\ &\quad + \left(1 - p^\pi_{G_0}(b) - p^\pi_{C_1}(b)\right)\mathbb{E} \\ &\quad \times \left[V^\pi(b') + \Theta'\middle| b, \pi(b), s \notin \mathcal{G}, b' \notin \mathcal{B}_C\right] - \Theta \end{aligned}$$

where $p^\pi_{G_n}(b)$ and $p^\pi_{C_n}(b)$ are the probabilities of reaching a goal or collision state *for the first time* after executing n actions following a policy π from the belief state b. That is, $p^\pi_{G_0}(b) = \Pr(s \in \mathcal{G}|b)$ and $p^\pi_{C_1}(b) = \Pr(b' \in \mathcal{B}_C|b, \pi(b)) = \Pr(s' \in \mathcal{S}_C|b, \pi(b))$. By developing the above expression for $V^\pi(b')$, and then to $V^\pi(b'')$ and future belief states, the value function can be written as a linear function of the collision penalty K as below.

$$V^\pi(b) + \Theta = \left(\sum_{n=1}^{N} p^\pi_{C_n}(b)\right) K + \left(\sum_{n=1}^{N} p^\pi_{G_{n-1}}(b)\right) \Theta^\pi_{G_N}(b)$$

$$+\left(1-\sum_{n=1}^{N}(p_{C_n}^{\pi}(b)+p_{G_{n-1}}^{\pi}(b))\right)\cdot$$
$$\mathbb{E}\Big[V^{\pi}(b^{(N)})+\Theta^{(N)}\Big|b,\pi,\{s,s',\ldots,s^{(N-1)}\}\notin\mathcal{G},$$
$$\{b',b'',\ldots,b^{(N)}\}\notin\mathcal{B}_C\Big] \tag{19}$$

where $\Theta_{G_n}^{\pi}(b)$ is the average goal flight time given that the goal is reached within n actions following a policy π from the belief state b.

Let us now consider the value function at the initial belief state b_0 ($\notin \mathcal{B}_C$), where $\Theta_0 = 0$. For all policies whose planning objective is to find a flight path to the goal states, the last term of (19) goes to zero as the iteration number N increases towards infinity. In result, the optimal value function at b_0 can be written as a sum of the collision penalty K and the average goal flight time $\Theta_G^{\pi}(b_0)$, weighed with the total collision and goal probabilities.

$$V^{\pi}(b_0)=\left(\sum_{n=1}^{\infty}p_{C_n}^{\pi}(b_0)\right)K+\left(\sum_{n=1}^{\infty}p_{G_{n-1}}^{\pi}(b_0)\right)\Theta_{G_\infty}^{\pi}(b_0)=p_C^{\pi}(b_0)K$$
$$+p_G^{\pi}(b_0)\Theta_G^{\pi}(b_0) \tag{20}$$

with $p_C^{\pi}(b_0)+p_G^{\pi}(b_0)=1$.

3.2. Penalty value determination for maximum allowable collision risk

Based on this nice linear relation (20), this paper proposes a method to determine a value for the collision penalty K such that the corresponding optimal policy π^* will satisfy a given maximum allowable collision risk, i.e., $p_C^{\pi^*}(b_0)\le p_{thd}$.

First, let us introduce two extreme policies, which optimize different criteria for the same PO-SSP problem but without the risk constraint. The first one is the safest policy π_S which prioritizes the vehicle safety over the mission efficiency, like the approaches in [36,37]. This safest policy corresponds to the optimal policy for our MOMDP when the collision penalty K is sufficiently large. Another extreme policy is the path-efficient policy, denoted by π_E, which on the contrary minimizes the goal time disregarding the collision risk. In addition, let us consider an artificial collision policy π_C which always brings the vehicle into an immediate collision from any state.

Then, it is obvious that the following inequalities are satisfied among the four different policies π_C, π_S, π_E and π^*.

$$0<p_C^{\pi_S}(b_0)\le p_C^{\pi^*}(b_0)\le p_C^{\pi_E}(b_0)<p_C^{\pi_C}(b_0)=1 \tag{21}$$

$$0<\Theta_G^{\pi_E}(b_0)\le\Theta_G^{\pi^*}(b_0)\le\Theta_G^{\pi_S}(b_0)<\infty \tag{22}$$

When $\Theta_G^{\pi_E}(b_0)=\Theta_G^{\pi_S}(b_0)$, the safest and path-efficient policies coincide and give the best optimal policy both in terms of the goal flight time and the collision risk. So there is no interest to solve the safety-constrained path planning. Hence, we consider the cases when $\Theta_G^{\pi_E}(b_0)<\Theta_G^{\pi_S}(b_0)$ and $p_C^{\pi_S}(b_0)<p_C^{\pi_E}(b_0)$. The above relations (21), (22) allow us to establish the following theorem.

Theorem 1. *The optimal policy π^*(16) to the MOMDP problem defined in Section 2 with the cost function (13) satisfies a user-defined safety requirement, $p_C^{\pi^*}(b_0)\le p_{thd}$ for any $p_{thd}>p_C^{\pi_S}(b_0)$, when the collision penalty value is given by*

$$K^*=\frac{p_G^{\pi_S}(b_0)\Theta_G^{\pi_S}(b_0)-(1-p_{thd})\Theta_G^{\pi_E}(b_0)}{p_{thd}-p_C^{\pi_S}(b_0)} \tag{23}$$

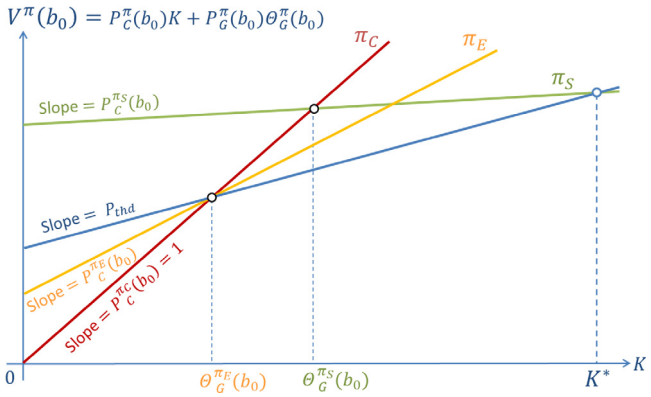


Fig. 5. The value function $V^{\pi}(b_0)$ for the different policies versus the collision penalty K. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Proof of Theorem 1. Fig. 5 plots the value function at the initial belief state b_0 for the two extreme policies, π_S (in green) and π_E (in yellow), versus the collision penalty value K. Following Eq. (20), each policy can be represented as a line with a slope of the total collision probability $p_C^{\pi}(b_0)$. The virtual collision policy π_C (in red) can be represented as a line from the origin with a slope $p_C^{\pi_C}(b_0)=1$. Since it is a line of $V^{\pi}(b_0)=K$, every policy line intersects with it at $K=\Theta_G^{\pi}(b_0)$.

When $K=K^*$ given in (23), from the relations $\Theta_G^{\pi_E}(b_0)<\Theta_G^{\pi_S}(b_0)$ and $p_C^{\pi_S}(b_0)<p_{thd}$,

$$K=K^*=\Theta_G^{\pi_S}(b_0)+\frac{1-p_{thd}}{p_{thd}-p_C^{\pi_S}(b_0)}(\Theta_G^{\pi_S}(b_0)-\Theta_G^{\pi_E}(b_0))\ge\Theta_G^{\pi_S}(b_0)$$

It means that $K=K^*$ lies after the safest policy line (in green) crosses the collision policy line (in red). This ensures that the collision policy π_C can never be optimal in terms of our optimization criteria. Given $K\ge\Theta_G^{\pi_S}(b_0)\ge\Theta_G^{\pi^*}(b_0)$, the optimal policy π^* will meet the safety requirement ($p_C^{\pi^*}(b_0)\le p_{thd}$) if the following inequality holds.

$$V^*(b_0)=p_C^{\pi^*}(b_0)K+(1-p_C^{\pi^*}(b_0))\Theta_G^{\pi^*}(b_0)$$
$$\le p_{thd}K+(1-p_{thd})\Theta_G^{\pi^*}(b_0) \tag{24}$$

Since the path-efficient policy π_E minimizes the goal flight time without caring the collision risk, the right hand side of (24) is lower-bounded by $p_{thd}K+(1-p_{thd})\Theta_G^{\pi_E}(b_0)$, which is drawn in blue line in Fig. 5. Thus, if $V^*(b_0)\le p_{thd}K+(1-p_{thd})\Theta_G^{\pi_E}(b_0)$, (24) satisfies. Besides, from the optimality condition, we have $V^*(b_0)\le V^{\pi_S}(b_0)=p_C^{\pi_S}(b_0)K+(1-p_C^{\pi_S}(b_0))\Theta_G^{\pi_S}(b_0)$. As seen in Fig. 5, the blue and green lines intersect at $K=K^*$. Consequently, the inequality (24) satisfies automatically when $K=K^*$, and hence the optimal policy π^* meets the safety requirement $p_C^{\pi^*}(b_0)\le p_{thd}$.

This theorem gives a guarantee of not violating a user-defined maximum allowable collision probability to the optimal policy solution of the unconstrained PO-SSP problem. However, the PO-SSP problem (even risk-unconstrained one) is not always mathematically tractable to derive the exact optimal policy π^*. Many POMDP solvers only approach to it. The theorem below shows a condition on the value function of a (non-optimal) policy π to guarantee the safety requirement.

Theorem 2. *A policy π to the MOMDP problem defined in Section 2 with the cost function (13) satisfies a user-defined safety requirement, $p_C^{\pi}(b_0)\le p_{thd}$ for any $p_{thd}>p_C^{\pi_S}(b_0)$, when the collision*

penalty value is given by (23) and its value function at the initial belief state satisfies $V^{\pi}(b_0) \leq V^{\pi_S}(b_0)$.

Proof of Theorem 2. From the definition of the path-efficient policy, $\Theta_G^{\pi}(b_0) \geq \Theta_G^{\pi_E}(b_0)$ for any policy π. Hence,

$$V^{\pi}(b_0) = p_C^{\pi}(b_0)K + (1 - p_C^{\pi}(b_0))\Theta_G^{\pi}(b_0) \geq p_C^{\pi}(b_0)K + (1 - p_C^{\pi}(b_0))\Theta_G^{\pi_E}(b_0)$$

At the same time, from (23) and the condition $V^{\pi}(b_0) \leq V^{\pi_S}(b_0)$, the following satisfies.

$$V^{\pi}(b_0) \leq V^{\pi_S}(b_0) = p_C^{\pi_S}(b_0)K + (1 - p_C^{\pi_S}(b_0))\Theta_G^{\pi_S}(b_0) = p_{thd}K + (1 - p_{thd})\Theta_G^{\pi_E}(b_0)$$

By combining the two inequalities, we obtain $(p_C^{\pi}(b_0) - p_{thd})(K - \Theta_G^{\pi_E}(b_0)) \leq 0$. Recall that $K = K^* \geq \Theta_G^{\pi_S}(b_0) > \Theta_G^{\pi_E}(b_0)$. Then the term $(K - \Theta_G^{\pi_E}(b_0))$ in the above inequality becomes strictly positive, and hence the safety requirement $p_C^{\pi}(b_0) \leq p_{thd}$ satisfies.

4. POMCP-GO algorithm

Solving a POMDP problem – or even a MOMDP problem – is not a trivial task. The process of updating the belief state is challenging, in particular in continuous state/action/observation spaces (e.g. real-world problems). Furthermore, in our MOMDP model, when applying (12), the belief state is deformed by the grid-based probability distribution $\Pr(s'_v|s_v, a, s'_h)$ and can no longer be represented as a Gaussian. Even if an approximation of such a Gaussian function could be learnt, it would be computationally expensive. Moreover, the computation of $\Pr(s_v|b, a)$, which is necessary to the value approximation, is also a time consuming step. Contrary to solvers such as SARSOP [20], HSVI [39] and RDTP-bel [26], Monte-Carlo Tree Search (MCTS)-based approaches, like Partially Observable Monte-Carlo Planning (POMCP) algorithm [27] and variants [40,41], are based on sampling and do not require to explicitly update the belief state in each decision step. Therefore, such approaches become a promising alternative to solve our planning problem.

This paper proposes a goal-oriented variant of the POMCP algorithm, named POMCP-GO. As shown later in Section 5, it enables to accelerate the convergence of the value, and thus to approach faster a promising path policy when compared to the classical version of POMCP. In the following, the classical POMCP algorithm [27] is recalled in the first subsection, followed by a presentation of the proposed POMCP-GO algorithm.

4.1. POMCP algorithm

POMCP is an *online* MCTS algorithm for partially observable environments [27]. It samples a state s from the initial belief state b_0 (root node) and simulates action–observation sequences (trials) in order to evaluate actions while constructing a tree of history (belief) nodes. The trial procedure is repeated during a short fixed time budget. Then, the best current action is applied and an observation is received, allowing to follow the related branch in the tree and to update the root node. As POMCP works online, those steps are performed along with the real action execution and observation until the mission ends.

Each tree node h represents a history of action–observation pairs from the initial belief state. Rather than updating the belief state after each action–observation pair, POMCP keeps in memory the number of times a node was explored $N(h)$ and the number of times a given action a was chosen $N(ha)$ in this node. This trick allows to approximate the Q-value of a belief state by $Q(h, a)$, which is the mean return from all trials started from the history h when action a was selected. Note this approximation differs from the classical definition of Q-value given in (17). This may incur a well-known bias [42] to the Q-value approximation that tends to decrease as the number of trials increases.

During planning, POMCP relies on the Upper Confidence Bounds (UCB1) action selection strategy [43] to deal with the exploration–exploitation dilemma [44].

$$\bar{a}_{\text{UCB}} = \arg\min_{a \in \mathcal{A}} \left\{ Q(h, a) - c \sqrt{\frac{\log N(h)}{N(ha)}} \right\} \tag{25}$$

While the first term in (25) vouches for the exploitation of the previously visited choice with the lowest cost values, the second encourages the exploration of undiscovered nodes in order to avoid falling into a local optimum. The larger the exploration coefficient $c > 0$ is, the more the exploration is prioritized over the exploitation. Hence the value of c directly influences the algorithm to perform either a breath-first or a depth-first tree search. If a leaf node is reached, a *rollout* procedure is performed in order to have an initial approximation of the value for this leaf node. Then, the algorithm updates the Q-value of all nodes visited during the trial by *back propagation* starting from the leaf node. Interested readers are invited to see [27,40,41] for more details.

4.2. POMCP-GO algorithm

The POMCP-GO algorithm, proposed in this paper and presented in Algorithm 1, is an offline goal-oriented variant of the POMCP algorithm for our PO-SSP problem. The main differences from the original POMCP are hereafter listed.

Algorithm 1: POMCP-GO

Function `POMCP-GO(`b_0`)`**:**
- $h = b_0 = (s_{v0}, b_{h0})$
- **while** $nbTrial < nb_{max}$ **do**
  - $s_{h0} \sim b_{h0}$
  - `Trial(`h, s_{v0}, s_{h0}`)`
  - $nbTrial += 1$
- return T^*, where $\forall h \in T^*, \pi^*(h) \leftarrow \arg\min_{a \in \mathcal{A}} Q(h, a)$

Function `Trial(`h, s_v, s_h`)`**:**
- **if** $(s_v, s_h) \in \mathcal{S}_T = \mathcal{G} \cup \{s_T\}$ **then**
  - return 0
- **if** $s_v \in \mathcal{S}_{v_C}$ $(F_C == 1)$ **then**
  - return $K - \Theta$, with $\Theta \in s_v$
- **if** $h \notin T$ **then**
  - **for** $a \in \mathcal{A}$ **do**
    - Creating ha node
    - $T(ha) \leftarrow (N_{init}(ha), Q_{init}(h, a), \emptyset)$
- $\bar{a} \leftarrow \bar{a}_{UCB}$ following Eq. (25)
- $\left(s'_v, s'_h, C(s_v, s_h, \bar{a})\right) \sim$ `TransitionModel(`s_v, s_h, $\bar{a}$`)`, cf. Section 2.3.2
- Creating hao node (if necessary) with $a = \bar{a}$, $o = s'_v$
- $Q(h, \bar{a})' \leftarrow C(s_v, s_h, \bar{a})$ + `Trial(`hao, s'_v, s'_h`)`
- $N(h) \leftarrow N(h) + 1$
- $N(h\bar{a}) \leftarrow N(h\bar{a}) + 1$
- $Q(h, \bar{a}) \leftarrow Q(h, \bar{a}) + \frac{Q(h,\bar{a})' - Q(h,\bar{a})}{N(h\bar{a})}$
- return $Q(h, \bar{a})'$

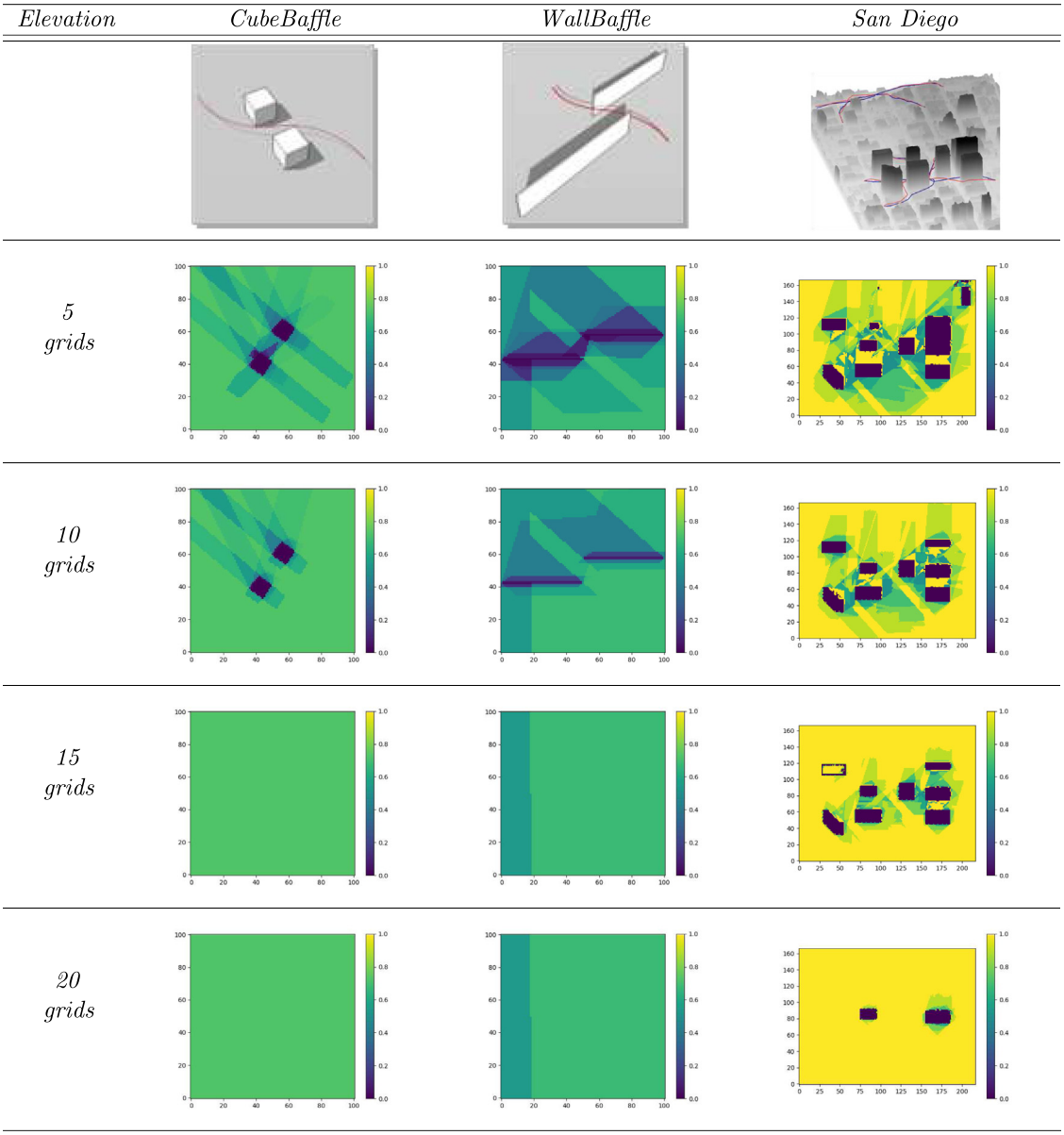


Fig. 6. Benchmark maps from [38] and the availability maps for different elevations. These probability maps were computed thanks to the Oktal-SE GPS simulator available at ONERA and the technique explained in Section 1.1 based on the GPS-PDOP maps obtained.

- In the classical POMCP algorithm, a trial procedure simulates the action–observation sequences until reaching a leaf node. In the POMCP-GO algorithm, these sequences end only when an absorbing state $s \in \mathcal{S}_\mathcal{T} = \mathcal{G} \cup \{s_T\}$ (either goal or collision) is reached (Line 9 in Alg. 1), which forces a depth-first search as proposed in [26]. Note it is no more only the UCB1 coefficient that influences the type of the search (i.e., breath- or depth-first), although it still plays a similar role as in POMCP for pushing the algorithm to play different actions (i.e. explore) time to time.
- Instead of performing a rollout procedure, POMCP-GO initializes the Q-value of a newly created node by a pre-computed heuristic value ($Q_{init}(h, a)$ in line 16). More precisely, it gives an optimistic estimation of the flight time based on the Dijkstra algorithm [45] performed on a grid obstacle map without considering uncertainties nor vehicle GNC model. This heuristic-based value initialization gives a more informative value approximation than a rollout policy resulted from random sampling of actions.
- Finally, after a specified number of trials (or a time budget), POMCP-GO returns the best policy tree computed (Line 7 in Alg. 1), while POMCP returns only the best action for the current root node. This difference is more related to the fact that POMCP-GO is an *offline* variant of POMCP algorithm. Nonetheless, it is straightforward to adapt POMCP-GO to *online* planning.

As the path cost is back propagated only when a terminal state is encountered during policy optimization, the POMCP-GO takes more computation time than the POMCP for the same number of trials. However, for the same time budget, the POMCP-GO algorithm is expected to be able to provide a qualitatively better policy for our planning problem, given that longer paths should help to identify collision risk faster and so to favor alternative paths earlier in optimization.

5. Experiment results

The approach proposed in this paper was evaluated in two steps.

Firstly, the performance of the POMCP-GO algorithm is compared with the state-of-the-art POMCP algorithm in terms of policy value convergence and mission success rate in two benchmark problems.

Secondly, the collision penalty determination method for an user-defined maximum allowable collision risk (Section 3) is empirically demonstrated in two other benchmark problems.

5.1. Experiment set-up

Environment maps. The evaluation uses environment maps available in the benchmark problems of UAV obstacle field navigation [38]: *WallBaffle*, *CubeBaffle*, and a digital elevation map of *San Diego* downtown (Fig. 6). The two first simple obstacle maps are represented in $100 \times 100 \times 20$ grids of 2 m on each side. The *San Diego* map has $217 \times 167 \times 21$ grids of 4 m. Thus the maximum altitude considered is 84 m for *San Diego* map, and 40 m for the other benchmark maps. For each environment map, a GPS availability probability map is generated from the PDOP values calculated by using the GPS simulator as explained in Section 2.1. Fig. 6 shows the GPS availability probability maps at different elevations. Obviously, the GPS becomes more available at higher altitude with less satellite occlusion.

Initial conditions. The initial mean position is set to $\boldsymbol{X}_0 = [50, 20, 5]$ for the *WallBaffle* and *San Diego* maps, and $[60, 40, 5]$ (in grids) for *CubeBaffle*, along with the zero-mean velocity and accelerometer bias, giving $\boldsymbol{x}_0 = [\boldsymbol{X}_0, \boldsymbol{0}, \boldsymbol{0}]$. The initial belief state is defined such as $b_0 = \left(s_{v0}, b_{h0} = \mathcal{N}(\boldsymbol{x}_0, \sum_0 = P_0)\right)$, where $s_{v0} = (F_{GNSS_0} = 1, F_{C_0} = 0, P_0, \Theta_0 = 0)$.

Goal states. The goal position is set to $\boldsymbol{X}_{\mathcal{G}} = [50, 80, 5]$ for the *WallBaffle* and *CubeBaffle* maps, and $[100, 70, 5]$ (in grids) for the *San Diego* map. The bounded region $\mathcal{G}_X$ is defined by a cube of 3-grid edge length centered at $\boldsymbol{X}_{\mathcal{G}}$.

Actions. Two sets of actions are considered: $\mathcal{A}_2$ is composed of only 4 desired horizontal velocity directions (North, South, East, West) with a constant speed $V_{ref} = 2.2$ m/s, and $\mathcal{A}_3$ is of 10 directions including 8 radial directions in the 2D horizontal plane, plus up and down. While the first set $\mathcal{A}_2$ was used for the *CubeBaffle* map, $\mathcal{A}_3$ was considered for the *WallBaffle* and *San Diego* maps.

GNC parameters. The parameters used in the GNC module (Section 2.2) are the followings: a sampling time $\Delta t = 0.4$ s, the initial error covariance $P_0 = \text{diag}(1\text{ m}, 1\text{ m}, 2\text{ m}, 0.1\text{ m/s}, 0.1\text{ m/s}, 0.2\text{ m/s}, 0.1\text{ m/s}^2, 0.1\text{ m/s}^2, 0.1\text{ m/s}^2)^2$,[4] the process noise covariance $Q = \text{diag}(0\text{ m}, 0\text{ m}, 0\text{ m}, 0\text{ m/s}, 0\text{ m/s}, 0\text{ m/s}, 0.2\text{ m/s}^2, 0.2\text{ m/s}^2, 0.2\text{ m/s}^2)^2$, the IMU acceleration noise covariance $R_a = \text{diag}(0.1\text{ m/s}^2, 0.1\text{ m/s}^2, 0.1\text{ m/s}^2)^2$, the GNSS measurement noise covariance $R_{GNSS} = \text{diag}(1\text{ m}, 1\text{ m}, 1\text{ m}, 0.1\text{ m/s}, 0.1\text{ m/s}, 0.1\text{ m/s})^2$, and the control gain for the velocity tracking guidance $K_d = 0.44$.

Planning model and algorithm parameters. In the planning model, the action execution time is set to $\Delta T_a = N_a \Delta t = 5\Delta t = 2$ s for $\forall a \in \mathcal{A}$. A large collision cost parameter $K = 450$ was used in the first test, and also in the second test to derive the safest policy π_S. Then in the second test, a value of K^* was explicitly computed for each study case according to (23). The following algorithm parameters were used: the UCB1 coefficient $c = 0.222 \times K$, and the total number of trials $nb_{max} = 10^5$. The POMCP rollout policy used to initialize the value of leaf nodes was replaced by the heuristic value obtained from the relaxed Dijkstra solution as done in the POMCP-GO algorithm.

Evaluation metrics. The evaluation was made for 5 optimization runs, since our algorithm is based on random Monte Carlo sampling. The optimized initial belief state value $V^{opt}(b_0)$ is recorded at every 5000 trials during optimization, and its evolution is analyzed in order to check the policy value convergence. Then, a current best policy obtained after every 5000 trials is evaluated by performing 1000 simulations. The initial belief state value $V^{sim}(b_0)$, the goal probability $P_G^{\pi}(b_0)$, and the average goal flight time $\Theta_G^{\pi}(b_0)$ resulted from the simulations are considered as evaluation metrics. Note that the resulting $V^{sim}(b_0)$ is the average cost for all trajectories executed during the simulation following the current best policy, which differs from $V^{opt}(b_0)$ obtained after the optimization due to the Q-value approximation bias.

5.2. Planning performance comparison : POMCP-GO vs. POMCP algorithms

First, the simple *CubeBaffle* and *WallBaffle* maps are used to compare the planning performance of the proposed POMCP-GO algorithm and the state-of-the-art POMCP. The comparative study was made only with an offline version of POMCP, since the particularities of our planning model hinder the application of most of the other POMDP solvers which are not based on Monte-Carlo sampling.[5] The recent variants of POMCP explore new action selection strategies, or Q-value update functions. Given that POMCP-GO introduces a new trial mechanism (goal-oriented) and a different node initialization that favors in-depth trials, those POMCP variants were excluded from our experiments. The presented analysis focuses on demonstrating how the goal-oriented mechanism improves the value convergence.

5.2.1. Cubebaffle map with $K = \bar{K} = 450$

Fig. 7 shows the comparative results for the *CubeBaffle* example. The evolution of $V^{opt}(b_0)$ (in the first column of the figure) demonstrates that the both algorithms converge after 10^5 trials. Although the value reached by POMCP is lower than that by POMCP-GO (69 against 96), the average value observed during simulations $V^{sim}(b_0)$ (in the second column) for POMCP-GO is lower than that for POMCP (88 against 113). This difference can be explained by the remaining bias on the optimized value. Since the POMCP-GO algorithm continues a trial until reaching an absorbing state (goal or collision), its $V^{opt}(b_0)$ tends to be always greater than the simulation average $V^{sim}(b_0)$. It is not the case for POMCP as it back-propagates the value of a trial once reaching a leaf node whose Q-value is initialized with the optimistic heuristic. The planning performance advantage of POMCP-GO is also affirmed by the mission success rate (in the third column — 85% for POMCP against 96% for POMCP-GO).

The POMCP policies result in shorter goal flight time in average than POMCP-GO (in the fourth column on Fig. 7). As illustrated in Fig. 8, the POMCP policies generate trajectories flying between the two cube obstacles favoring the flight time minimization, while the POMCP-GO policies make a detour to increase the goal probability. Thanks to its depth-first search, POMCP-GO is able to deviate from the heuristic policy paths to improve the value much earlier (w.r.t. the number of trials). However, the computation

[4] $\text{diag}(x_1, x_2, \ldots)$ represents a diagonal matrix with the diagonal elements of $x_1, x_2, \ldots$.

[5] In [46], the authors actually compared the planning performance with the heuristic policy following the shortest path obtained by Dijkstra algorithm over a discretized position space. However, it is not a fair comparison as the heuristic policy does not take into account the collision risk issued from the sensor availability and the navigation uncertainty.

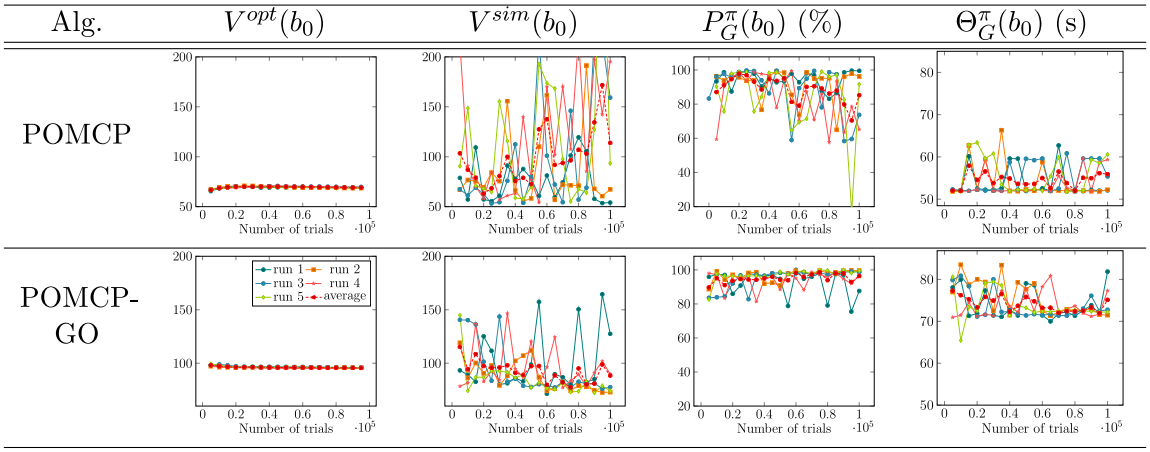
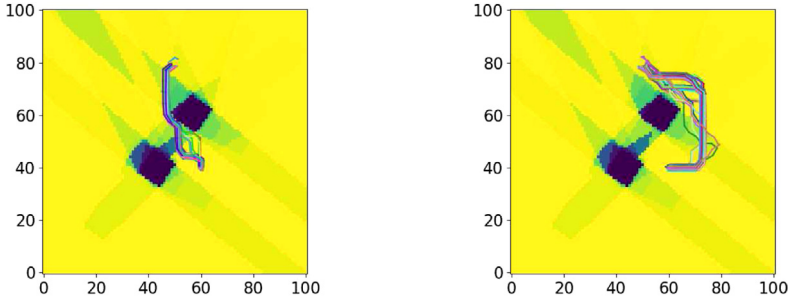


Fig. 7. POMCP and POMCP-GO performance comparison with the A_2 action set for the *CubeBaffle*.



(a) POMCP (b) POMCP-GO

Fig. 8. Path examples obtained with POMCP (run 1 - a) and POMCP-GO (run 2 - b) for the *CubeBaffle*.

time for the same number of trials is 50% longer than that of POMCP in our example.

In conclusion, the trade-off between policy computation time and policy quality must be carefully analyzed. When comparing the evaluation results of a POMCP-GO policy obtained after 6×10^4 trials and a POMCP policy after 10×10^4 trials (after almost the same computation time), the POMCP-GO policy still presents the lower simulated value, implying "more" optimal.

5.2.2. WallBaffle map with $K = \bar{K} = 450$

Fig. 9 compares the results of POMCP and POMCP-GO on the *WallBaffle* map. In this example, a large gap in the optimized and simulated values (49 against 213) is observed for the POMCP case. The optimized value is similar to the one obtained with a heuristic policy. This implies that, in this example, the number of trials was not sufficient for the POMCP algorithm to search in depth for deviating from the heuristic policy. The simulation results show the poor goal probability (69%) of the POMCP policy, while POMCP-GO attains 99.7%. This is due to a fact that, during simulations, the heuristic policy is applied when encountering a leaf (but not absorbing) node of the policy tree. Simulated path examples are illustrated in Fig. 10. The POMCP-GO policy finds a path to fly over the walls for exploiting more chance to have GPS (see Fig. 6), while the POMCP policy still tries to fly between them where GPS is less-likely available. These results confirm an advantage of the proposed POMCP-GO algorithm over the classical POMCP in solving our safe path planning problem.

5.3. Results of the safety-constrained path planning

In this section, the *WallBaffle* and *San Diego* maps are used to empirically demonstrate the method proposed in Section 3 to compute the collision penalty value K for a given safety requirement. Moreover, the impact on the policy behavior (e.g. paths) is also evaluated for different values of the collision probability threshold p_{thd}.

5.3.1. WallBaffle map with $p_{thd} = 10\%$ and 40%

Given the previous results from Section 5.2.2, one can consider one of the resulting policies that reached a 100% of success as the *safe policy* π_S, defined in Section 3. It gives the average goal flight time $\Theta_G^{\pi_S}(b_0) = 75$ s with the goal probability $p_G^{\pi_S}(b_0) = 1$ (cf. Fig. 9). The path-efficient policy, π_E, can be approximated by the heuristic policy, which is the shortest-path solution of the relaxed problem (on a 3D grid map without considering uncertainty and vehicle motion dynamics) obtained by the Dijsktra algorihtm. It gives the average goal flight time $\Theta_G^{\pi_S}(b_0) = 61$ s. Then, one can calculate the collision penalty value K^* for a given collision risk threshold p_{thd} according to Eq. (23). For example, we obtain $K^*_{0.1} = 201$ and $K^*_{0.4} = 96$ for $p_{thd} = 0.10$ and $p_{thd} = 0.40$ respectively.

The planning results of the cases of $K = \bar{K}, K^*_{0.1}$ and $K^*_{0.4}$ are compared in Fig. 11. When $p_{thd} = 0.10$, the optimized initial belief state value $V^{opt}(b_0)$ tends to 90 after 10^5 trials, while the initial belief state observed during simulations $V^{sim}(b_0)$ achieves

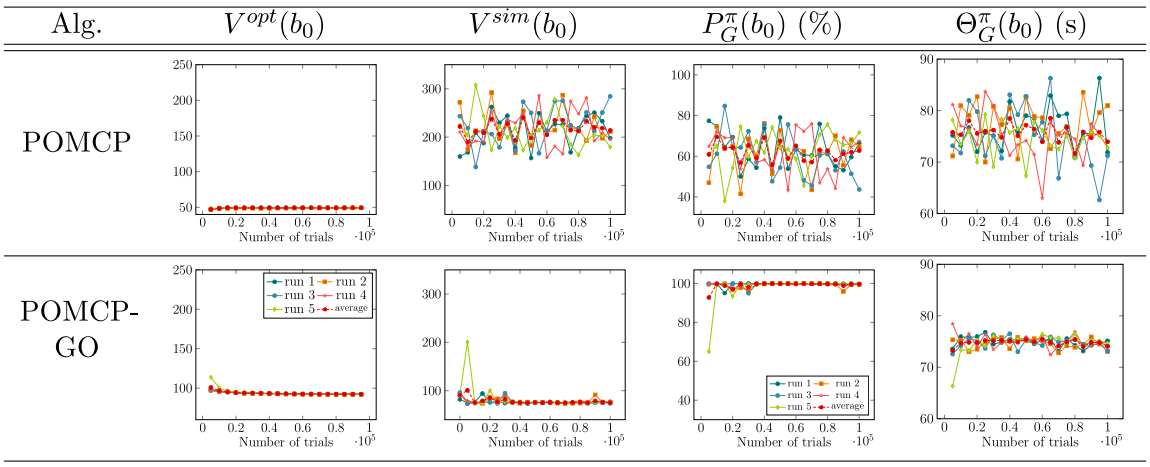


Fig. 9. POMCP and POMCP-GO performance comparison with the A_3 action set for the *WallBaffle*.

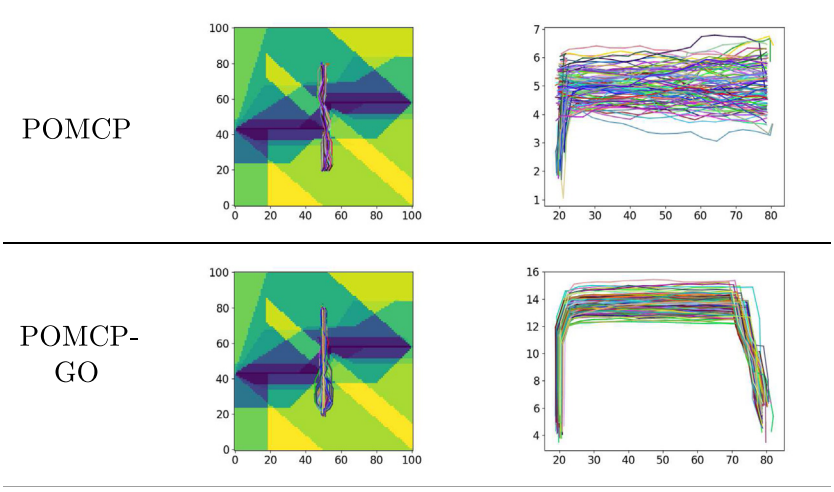


Fig. 10. Path examples obtained with POMCP and POMCP-GO for the *WallBaffle* map: (left) horizontal path and (right) altitude.

74 (except for Run 1), which is lower than $V^{\pi_S}(b_0) = 75$. Thus, Theorem 2 can guarantee that those computed policies will respect the given safety requirement. This is confirmed by the success rate result $p_G^{\pi}(b_0) = 99.9\%$. These policies have the average goal flight time (73 s) slightly shorter than $\Theta_G^{\pi_S}(b_0) = 75$ s. This is an expected result, as the decrease in the K value would put more priority on the mission *Efficiency* over *Safety*. The same analysis can be done for the results of $p_{thd} = 0.40$. These results demonstrate the interest of the proposed safety-constrained path planner, which computes a policy that decreases the flight time by admitting a certain risk of collision depending on the mission requirement.

5.3.2. San Diego map and $p_{thd} = 40\%$

The same approach of the collision penalty determination was applied to the *San Diego* map. Firstly, policies were computed with $K = \bar{K} = 450$ to determine the safest policy π_S. Its results are shown in the first column of Fig. 12. The last policy obtained after 10^5 trials in Run 4 (in pink) attaining 100% success and $\Theta_G^{\pi_S}(b_0) = 105$ s was chosen for π_S. The path-efficient policy π_E is approximated by the heuristic policy with $\Theta_G^{\pi_E}(b_0) = 82$ s. Then, imposing an allowable collision risk of $p_{thd} = 0.4$, the collision penalty value can be derived as $K^*(p_{thd} = 0.4) = K^*_{0.4} = 140$ according to (23).

The second column of Fig. 12 shows the results with this new value of $K = K^*_{0.4}$. While the optimized initial belief state value is slightly above $\Theta_G^{\pi_S}(b_0)$, the simulated value $V^{sim}(b_0)$ achieves lower and hence satisfies the condition of Theorem 2. The success rate of the computed policies after 10^5 trials is at least 65% (with Run 5), and 74% in average, respecting the 40% of maximum collision risk. In return of admitting this 40% collision risk, the average flight time was reduced from 105*s* to 83*s* in average.

Note that this San Diego map represents a realistic but challenging environment for the vehicle navigation, as there is more risk of losing the GPS availability than in the previous simple obstacle scenarios. Interestingly, the simulated paths for the policies with $K = \bar{K}$ and $K^*_{0.4}$ are quite different. While the safe policy chooses to fly over the building for favoring the GPS availability and the mission *Safety*, the policies with $p_{thd} = 0.4$ choose to stay at the initial low altitude and to fly between the buildings where GPS is less likely available – increasing the collision risk due to the larger path execution uncertainty – for favoring the mission *Efficiency*.

6. Conclusions and future work

This paper proposed a safe path planner for UAV autonomous operation in an urban environment, which takes into account the environment-dependent uncertain sensor availability and its influence on the path execution accuracy. It adopted a MOMDP model which incorporates the low-level GNC module in the planning task in order to propagate the UAV state uncertainty without

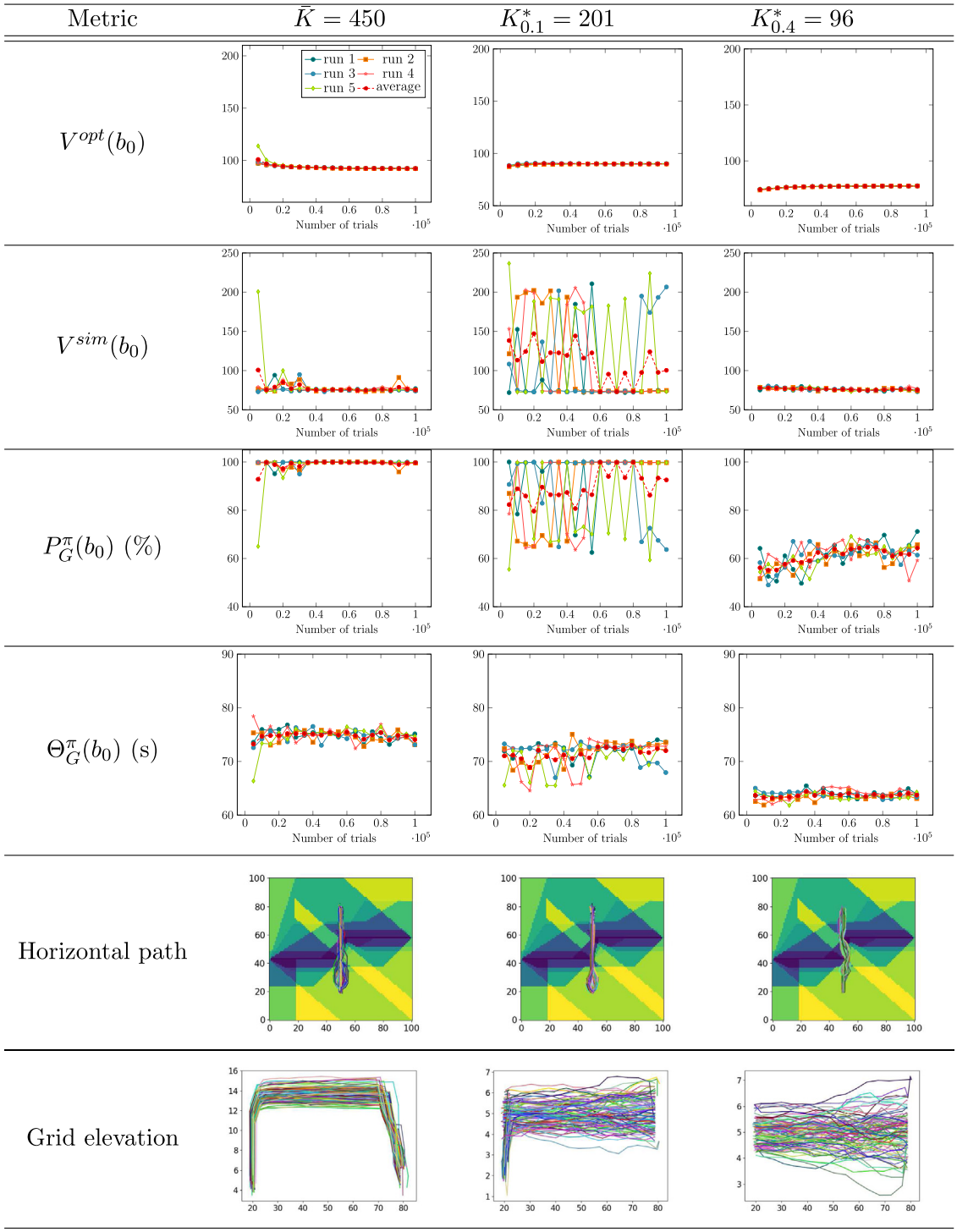


Fig. 11. POMCP-GO planning results for the *WallBaffle* benchmark for $\mathcal{A}_3$ action set and different collision penalty values.

ignoring the stochasticity in the sensor measurements and in the guidance commands. It also introduced a new cost function definition for the (PO-)SSPUDE problem, along with the method to determine the penalty value for a given safety requirement. Furthermore, a new POMDP solving algorithm, POMCP-GO, was proposed by extending the POMCP algorithm to goal-oriented version. The proposed safe path planner was evaluated on the simple and real obstacle maps. The results firstly showed the superior planning performance of our POMCP-GO algorithm over the classical POMCP. Then, we also proved the capability of our planner to compute a policy which respects the user-defined safety requirement without imposing it explicitly while planning.

Despite of the fact that MCTS-based algorithms are actually the best candidates to solve our complex MOMDP planning problem, the POMCP-GO algorithm still takes about few hours to converge (for the presented test cases), which is far from being able to run online. The perspectives of this work includes the following topics.

- Finding a new and more efficient heuristic function to initialize the action Q-value.

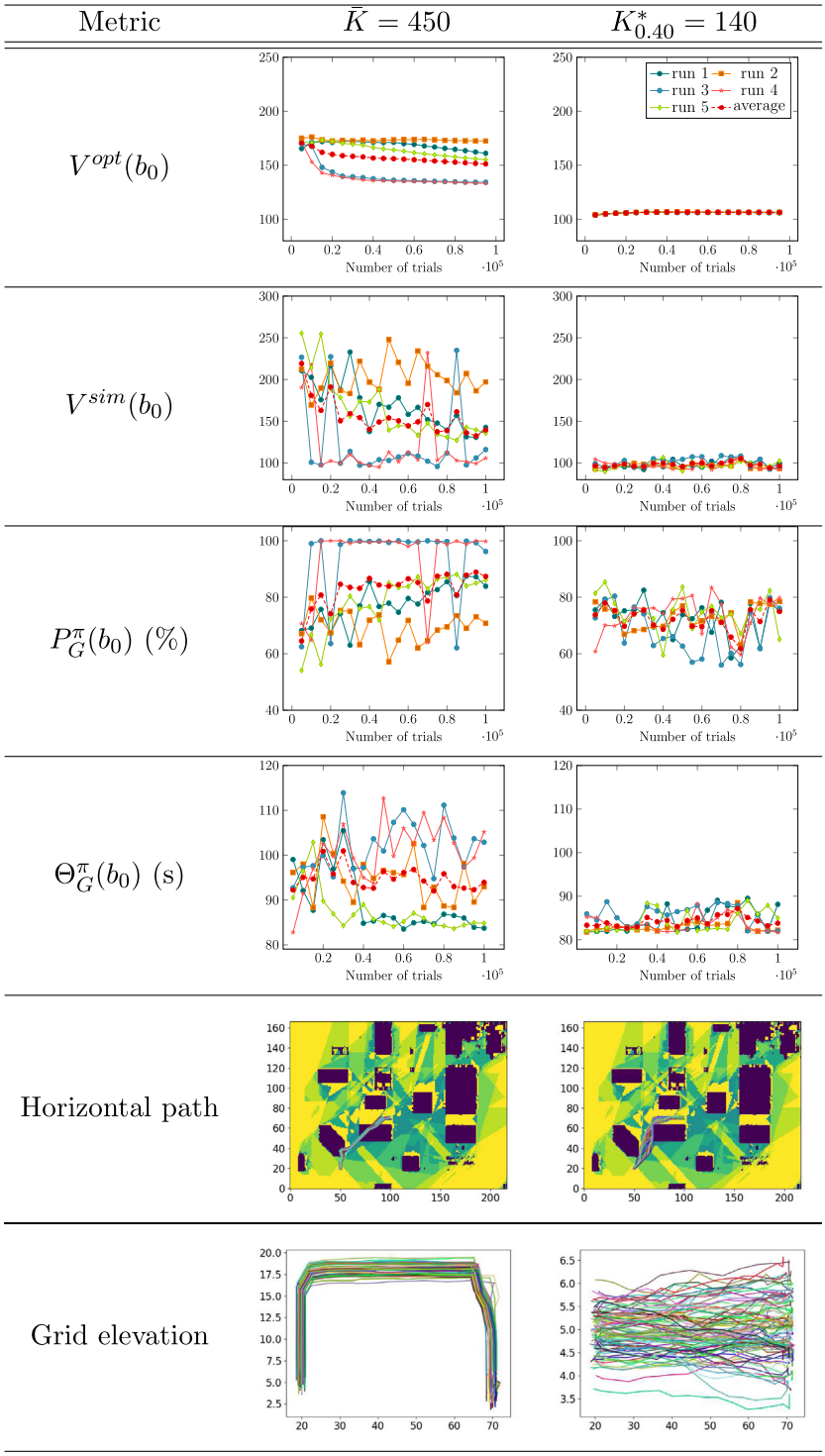


Fig. 12. POMCP-GO planning results for the *San Diego* benchmark with the $\mathcal{A}_3$ action set and different collision penalty values.

- Adapting the UCB1 exploration coefficient c in function of the environment complexity. The authors have already started to investigate it in [47].
- Introducing a debranching strategy which identifies useless branches of the decision tree and removing them to save the memory-use. We have actually applied an ad-hoc strategy to avoid the memory saturation, showing its efficiency. We would like to further explore this idea to establish a theoretically-proven debranching strategy.
- Decreasing the Q-value bias of POMCP and POMCP-GO by applying a different back-propagation strategy, as suggested in [42]. Such bias reduction should improve the convergence speed of the algorithm [47].
- Investigating original methods to clustering similar (belief) states to better initialize its Q-value, and also making the policy solution generalize to belief states (or nodes) not visited during the optimization process.
- Integrating the improved planning algorithm in the AMPLE framework [48], which develops a plan-while-executing

strategy. This framework ensure reactivity to action requests from the execution engine, while being able to constantly improve the policy by prioritizing future execution states and computing a partial but applicable policy for each of them.

With these planning performance improvements, our ultimate goal is to extend this work to online safe path planner and to implement and flight-test it on real UAV platforms for enabling their safe urban operations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Defense Innovation Agency (AID) of the French Ministry of Defense (research project CONCORDE No 2019 65 0090004707501).

The authors appreciate Dr. Jonathan Israel at Department of Electro Magnetism and Radar, ONERA. He has contributed to this work by simulating GPS signals in the benchmark environments and providing us with the 3D GPS-PDOP maps from which we have calculated the GPS availability probability maps.

References

[1] F. Kleijer, D. Odijk, E. Verbree, Prediction of GNSS availability and accuracy in urban environments case study schiphol airport, Location Based Services and TeleCartography II – from Sensor Fusion to Context Models, Springer, 2009, pp. 387–406.

[2] B. Mettler, Z. Kong, C. Goerzen, M. Whalley, Benchmarking of obstacle field navigation algorithms for autonomous helicopters, in: AHS Annual Forum, 2010.

[3] L. Blackmore, M. Ono, B.C. Williams, Chance-constrained optimal path planning with obstacles, Trans. Robot. 27 (6) (2011) 1080–1094.

[4] M. da Silva Arantes, C.F.M. Toledo, B.C. Williams, M. Ono, Collision-free encoding for chance-constrained nonconvex path planning, Trans. Robot. 35 (2) (2019) 433–448.

[5] R. He, S. Prentice, N. Roy, Planning in information space for a quadrotor helicopter in a GPS-denied environment, in: IEEE International Conference on Robotics and Automation, 2008.

[6] M.W. Achtelik, S. Lynen, S. Weiss, M. Chli, R. Siegwart, Motion- and uncertainty-aware path planning for micro aerial vehicles, J. Field Robotics 31 (4) (2014).

[7] Y. Watanabe, S. Dessus, P. Fabiani, Safe path planning with localization uncertainty for urban operation of VTOL UAV, in: AHS Annual Forum, 2014.

[8] G. Zhang, L.-T. Hsu, A new path planning algorithm using a GNSS localization error map for UAVs in an urban area, J. Intell. Robot. Syst. 94 (2019) 219—235.

[9] M. Maaref, Z.M. Kassas, Optimal GPS integrity-constrained path planning for ground vehicles, in: 2020 IEEE/ION Position, Location and Navigation Symposium (PLANS), 2020, pp. 655–660.

[10] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. 30 (7) (2011).

[11] S.J. Prentice, N. Roy, The belief roadmap: Efficient planning in belief space by factoring the covariance, Int. J. Robot. Res. 28 (11–12) (2009) 1448–1465.

[12] A. Bry, N. Roy, Rapidly-exploring random belief trees for motion planning under uncertainty, in: IEEE International Conference on Robotics and Automation, 2011.

[13] F. Causa, G. Fasano, M. Grassi, Multi-UAV path planning for autonomous missions in mixed GNSS coverage scenarios, Sensors 18 (12) (2018) 4188.

[14] R. Platt, R. Tedrake, L. Kaelbling, T. Lozano-Perez, Belief space planning assuming maximum likelihood observations, in: Robotics: Science and Systems, 2010.

[15] R.D. Smallwood, E.J. Sondik, The optimal control of partially observable Markov processes over a finite horizon, Oper. Res. 21 (5) (1973) 1071–1088.

[16] L.P. Kaelbling, M.L. Littman, A.R. Cassandra, Planning and acting in partially observable stochastic domains, Artif. Intell. 101 (1–2) (1998) 99–134.

[17] M.T. Spaan, N. Spaan, A point-based POMDP algorithm for robot planning, in: IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, 3, IEEE, 2004, pp. 2399–2404.

[18] J. Pineau, G. Gordon, S. Thrun, Anytime point-based approximations for large POMDPs, J. Artificial Intelligence Res. 27 (2006) 335–380.

[19] C.P.C. Chanel, F. Teichteil-Königsbuch, C. Lesire, Multi-target detection and recognition by uavs using online pomdps, in: Twenty-Seventh AAAI Conference on Artificial Intelligence, 2013.

[20] H. Kurniawati, D. Hsu, W.S. Lee, Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces., in: Robotics: Science and Systems, Vol. 2008, Zurich, Switzerland, 2008.

[21] S.C. Ong, S.W. Png, D. Hsu, W.S. Lee, Planning under uncertainty for robotic tasks with mixed observability, Int. J. Robot. Res. 29 (8) (2010) 1053–1068.

[22] A. Foka, P. Trahanias, Real-time hierarchical POMDPs for autonomous robot navigation, Robot. Auton. Syst. 55 (7) (2007) 561–571.

[23] M. Araya-López, V. Thomas, O. Buffet, F. Charpillet, A closer look at MOMDPs, in: 2010 22nd IEEE International Conference on Tools with Artificial Intelligence, 2, IEEE, 2010, pp. 197–204.

[24] A. akbar Agha-mohammadi, S. Chakravorty, N.M. Amato, FIRM:Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements, Int. J. Robot. Res. 33 (2) (2014).

[25] J. van den Berg, S. Patil, R. Alterovitz, Motion planning under uncertainty using iterative local optimization in belief space, Int. J. Robot. Res. 31 (11) (2012) 1263–1278.

[26] B. Bonet, H. Geffner, Solving pomdps: RTDP-bel versus point-based algorithms, in: Twenty-First International Joint Conference on Artificial Intelligence (ICAPS), 2009.

[27] D. Silver, J. Veness, Monte-Carlo Planning in large POMDPs, in: Advances in Neural Information Processing Systems, 2010, pp. 2164–2172.

[28] S.D. Patek, On partially observed stochastic shortest path problems, in: Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228), 5, IEEE, 2001, pp. 5050–5055.

[29] Mausam, A. Kolobov, Planning with Markov decision processes: An AI perspective, Synth. Lect. Artif. Intell. Mach. Learn. 6 (1) (2012) 1–210.

[30] D.P. Bertsekas, J.N. Tsitsiklis, An analysis of stochastic shortest path problems, Math. Oper. Res. 16 (3) (1991) 580–595.

[31] M. Steinmetz, J. Hoffmann, O. Buffet, Goal probability analysis in probabilistic planning: exploring and enhancing the state of the art, J. Artificial Intelligence Res. 57 (2016) 229–271.

[32] A. Shetty, G.X. Gao, Predicting state uncertainty for GNSS-based UAV path planning using stochastic reachability, in: Proceedings of the 32nd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2019), 2019, pp. 131–139.

[33] J. Hoey, P. Poupart, Solving POMDPs with continuous or large discrete observation spaces, in: International Joint Conference on Artificial Intelligence, 2005, pp. 1332–1338.

[34] F. Trevizan, S. Thiébaux, P. Santana, B. Williams, I-dual: solving constrained SSPs via heuristic search in the dual space, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 4954–4958.

[35] A. Undurti, J.P. How, An online algorithm for constrained POMDPs, in: 2010 IEEE International Conference on Robotics and Automation, IEEE, 2010, pp. 3966–3973.

[36] A. Kolobov, Mausam, D.S. Weld, A theory of goal-oriented MDPs with dead ends, in: Conference on Uncertainty in Artificial Intelligence (UAI), 2012.

[37] F. Teichteil-Königsbuch, Stochastic safest and shortest path problems, in: AAAI Conference on Artificial Intelligence, 2012.

[38] B. Mettler, Z. Kong, C. Goerzen, M. Whalley, Benchmarking of obstacle field navigation algorithms for autonomous helicopters, in: Annual Forum Proceedings-AHS International, Vol. 3, American Helicopter Society, 2010, pp. 1936–1953.

[39] T. Smith, R. Simmons, Heuristic search value iteration for POMDPs, in: The Conference on Uncertainty in Artificial Intelligence (UAI), 2004.

[40] A. Somani, N. Ye, D. Hsu, W.S. Lee, DESPOT: Online POMDP planning with regularization, in: Advances in Neural Information Processing Systems, 2013, pp. 1772–1780.

[41] Z.N. Sunberg, M.J. Kochenderfer, Online algorithms for POMDPs with continuous state, action, and observation spaces, in: M. de Weerdt, S. Koenig, G. Röger, M.T.J. Spaan (Eds.), Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, the Netherlands, June 24-29, 2018, AAAI Press, 2018, pp. 259–263.

[42] T. Keller, M. Helmert, Trial-based heuristic tree search for finite horizon MDPs, in: Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS), 2013.

[43] L. Kocsis, C. Szepesvári, Bandit based monte-carlo planning, in: European Conference on Machine Learning, Springer, 2006, pp. 282–293.
[44] T. Schulte, T. Keller, Balancing exploration and exploitation in classical planning, in: Seventh Annual Symposium on Combinatorial Search, 2014.
[45] E.W. Dijkstra, et al., A note on two problems in connexion with graphs, Numer. Math. 1 (1) (1959) 269–271.
[46] J.-A. Delamer, Y. Watanabe, C. P. Carvalho Chanel, Solving path planning problems in urban environments based on a priori sensors availabilities and execution error propagation, in: AIAA Scitech 2019 Forum, 2019, p. 2202.
[47] A.R. Carmo, J.-A. Delamer, Y. Watanabe, R. Ventura, C.P. Chanel, Entropy-based adaptive exploit-explore coefficient for Monte-Carlo path planning, in: Prestigious Applications of Intelligent Systems, 2020.
[48] C.P.C. Chanel, A. Albore, J. T'Hooft, C. Lesire, F. Teichteil-Königsbuch, Ample: an anytime planning and execution framework for dynamic and uncertain problems in robotics, Auton. Robots 43 (1) (2019) 37–62.

Jean-Alexis Delamer: received his Ph.D. in Robotics and Automation from the Université de Toulouse, Toulouse, France. He is currently a Postdoc with the School of Computing of Queen's University, Kingston, ON. Jean-Alexis's research interests are mainly focused on Markov decision processes, autonomous systems and robotics.

Yoko Watanabe: received her Ph.D. degree in Aerospace Engineering in 2008 from Georgia Institute of Technology. She joined the applied control research unit at ONERA/DTIS in 2008, where she has been working on the autonomous navigation and guidance systems design for aerial vehicles and their flight testing. Her research activity covers robust navigation and state estimation by multi-sensor fusion, safe path planning problems under uncertainty, and coordination control of multi drones. She was awarded ICAS John J. Green Award in 2020 for her contribution to coordinating the EU-Japan research cooperation in Aeronautics.

Caroline P.C. Chanel: received the B. Eng. degree in Automatic control from Pontifícia Universidade Católica do Rio Grande do Sul, and her M. Sc. in Automatic, Informatics and Decision Systems from Université de Toulouse III. She holds a Ph.D. in Automatic Control and Embedded Systems. Her Ph.D. received a special mention from ISAE-SUPAERO Foundation. Caroline was Amelia Earhart Fellow in 2011. Currently, she works at ISAE-SUPAERO as Associate Professor in the Aerospace vehicles Design and Control Department. Her research aims to develop decision models for path, perception and mission planning considering uncertainties and degraded functioning modes, as well as, decision models to drive the interaction between automated systems and human operators.