Navigation and guidance strategy online planning and execution for autonomous UAV

Ana Raquel Araújo Simão do Carmo
raquelcarmo@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2019

Abstract

Efficient path planning for autonomous Unmanned Aerial Vehicles (UAVs) in cluttered environments is a challenging decision-making problem under uncertainty. Recently, Mixed-Observability Markov Decision Processes (MOMDPs) have been proposed to solve such navigation problems with the goal of reducing computational effort. In this work, a Partially Observable Stochastic Shortest Path (PO-SSP) planning problem is modelled as a MOMDP. The resulting model enables the planner to deal with a priori probabilistic sensor availability and path execution error propagation, which depend on the navigation and guidance solution used. This approach results in a large and complex path planning problem to be solved. To address this issue, a goal-oriented variant of the Partially Observable Monte-Carlo Planning (POMCP), one of the fastest online state-of-the-art Monte-Carlo Tree Search algorithms for partially observable environments, is used. It relies on the Upper Confidence Bounds (UCB1) as an action selection strategy, which depends on a coefficient typically adjusted manually that requires an exhaustive search to find the most suitable value. This exhaustive search applied to such a complex planning problem may be extremely time consuming. Therefore, dynamic coefficients are proposed as a first contribution of this dissertation. A second contribution is to incorporate this navigation and guidance strategy in a planning-while-executing framework. The resulting structure provides strictly anytime solutions, while exploiting action execution time to anticipate and plan future states, minimizing the overall duration of the mission. Simulations performed in Gazebo allow to evaluate the performance of this structure.

Keywords: Path Planning under Uncertainty, Navigation and Guidance Strategy, Mixed-Observability Markov Decision Process, Concurrent Planning and Execution.

1. Introduction

Unmanned Aerial Vehicles (UAVs) constitute a research field that has been extensively explored in the last decade. Recent advances made it possible for an UAV to develop autonomous capabilities, that is, to perform tasks without requiring a human to control them. As a result, UAVs can nowadays be employed in a variety of operations, ranging from search and rescue, surveillance and inspection missions [2], to navigation missions in cluttered environments (e.g. urban areas) [14].

Autonomous navigation capabilities include functionalities embedded in a Guidance, Navigation and Control system (GNC) onboard the UAV system. The navigation module estimates the state of the vehicle based on the measurements of the onboard sensors. Besides the classical solution combining Inertial Navigation Systems (INS) with Global Positioning Systems (GPS) as a self-localization system, one can also rely on solutions based on 2D or 3D vision sensors: visual odometry, visual Simultaneous Localization and Mapping (SLAM), etc. [2] [14].

However, all of these solutions have their own limitations. An INS-only navigation solution quickly diverges due to accumulation of inertial measurement bias, so it is usually coupled with GPS. Yet, GPS signal can easily be masked or degraded due to occlusion in cluttered environments. Consequently, a UAV cannot sustain autonomous flight in the absence of GPS signal, if it is not backed up with alternative sensors that do not rely on GPS information. Nevertheless, visual SLAM and optical flow techniques require rich texture on the image. Thus it is of crucial importance to embed, in the UAV onboard flight system, different navigation solutions and their use in each phase of the mission to be decided according to the surrounding environment.

Fortunately, the performance of some sensors can be predicted by a priori knowledge about the environment. For instance, given a known satellite constellation and a position, it is possible to obtain information about the accuracy of a GPS measurement in the form of Position Dilution of Precision (PDOP) [4] [7]. This information can be very useful for path planning, as this allows the planner to anticipate an eventual degradation of the sensors' accuracy and its influence on the path execution error along the way, minimizing the collision risk [3].

Although extensive research exists on applications of these UAV localization and guidance approaches, only recent developments can be found on the decision of switching navigation modes among the ones available onboard as a function of the availability of the sensors that depend on the environment [1] [15]. Delamer et al. have explored the complete model of closed-loop vehicle motion with the functions of GNC in the decision-making process, while taking into account the sensors' availability, in a significant body of work, including ([4], [3]). Nevertheless, such developments were achieved under an offline configuration of the planning algorithm, which is not suitable for applications in real scenarios with environments as dynamic as urban areas.

Having that in consideration, the main objective of this work is to apply a generic planning-while-executing framework to deal with the continuous state space problem addressed in an real-time configuration onboard a UAV. In particular, to handle possible evolutions of the environment and of the UAV state during the mission execution, providing strictly anytime solutions. In such framework, a GNC transition model will be integrated in the decision-making process of the online planner, thereby allowing the propagation of the belief state according to the availability of the onboard sensors.

This paper is organized as follows: firstly a section on previous work briefly describes the GNC model used and formalizes the problem addressed as a Mixed-Observability Markov Decision Process (MOMDP). This is followed by a section on the offline approach, which presents the planning algorithm to solve the MOMDP problem in an offline configuration and our contribution on strategies for parameter tuning. After, the reader is led through the online approach, in which the planner is converted into an online configuration and a brief explanation is done of the planning-while-executing framework that will drive the planner. Simulation results illustrate the advantages and disadvantages of the proposed approach. Furthermore, the performance of this framework is also tested in a robot simulator, Gazebo. Lastly, conclusions and future work are discussed.

2. Previous work

Following the work developed in [3], the problem addressed aims at finding a navigation and guidance strategy for making UAVs reach a given destination safely (avoiding obstacles) and efficiently (minimum distance or time) in a cluttered environment under uncertainty, thus making it a Partially Observable Stochastic Shortest Path (PO-SSP) planning problem. It takes into account the availability of sensors that depend on the environment and, with that, decides on the navigation and guidance strategy to be used by the UAV. Given the nature of the state variables being considered, [3] make use of the MOMDP representation [10] to model this PO-SSP problem.

To illustrate this approach, this work focuses on the UAV model proposed by [3], where the GNC closed-loop system (Figure 1) is modelled as a transition function of the continuous vehicle state vector $\mathbf{x}$, which comprises the position, velocity and accelerometer bias ($\mathbf{x} = \begin{bmatrix} \mathcal{X}^T & \mathcal{V}^T & \beta_a^T \end{bmatrix}^T$). The navigation module relies on the Extended Kalman filter to estimate this state $\mathbf{x}$ and its error covariance $\mathbf{P}$ for a selected navigation mode. The guidance and control module then executes a selected path by using the navigation solution. The execution precision is given by the execution error covariance $\sum$, which may depend on $\mathbf{P}$.

The architecture of the overall planning system considers the (closed-loop) GNC model as part of the MOMDP planning problem. The planner will then consider GNC's closed-loop transitions to compute the possible evolutions of the system's state. The policy computed by the planner take as inputs the probability distribution over the current state, b_{s_h}, and a vector of booleans on sensors' availability, s_v, and returns an action a, which contains information on a reference velocity to pursue, $\mathcal{V}_{ref}$, and on the navigation mode to use. Afterwards, a new vector of sensors' availability s'_v is observed from the environment and the belief state update is performed. The new belief $b_a^{s'_h}$ is then used by the policy to define the next action to execute. Figure 1 illustrates the system's architecture.

A priori knowledge on the environment is assumed to be given as a set of probability grid maps of obstacles and availability of each of the sensors (Figure 2). These maps are used during the planning task to propagate the path execution uncertainty given the probabilistic sensor's availability, and then to evaluate obstacle collision risk.

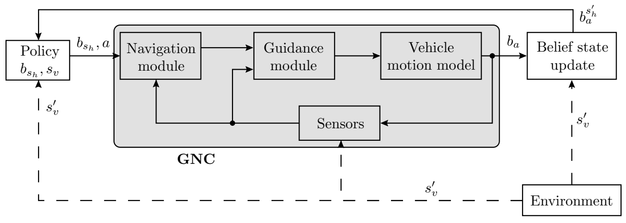
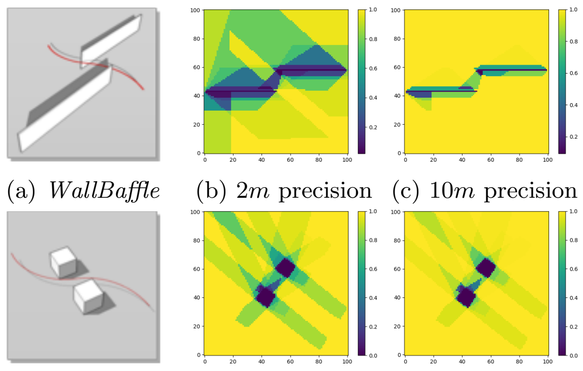


Figure 1: System architecture diagram [3].

2.1. MOMDP Model

Following [3], the PO-SSP path planning problem addressed is modelled as a MOMDP problem. Assuming that a vehicle always knows if a given sensor can be used or not in the current decision time step, then the sensor's availability is considered as a fully observable state variable of the model. On the other


(a) *WallBaffle* (b) $2m$ precision (c) $10m$ precision

(d) *CubeBaffle* (e) $1m$ precision (f) $2m$ precision

Figure 2: Obstacle maps [8] and examples of probability maps of GPS availability with different precision thresholds, given in *meters*.

hand, the vehicle state vector $\mathbf{x}$ is non-observable from the planning model point of view, as the only outputs from the GNC model are the localization and execution error covariances $\mathbf{P}$ and $\sum$.

The MOMDP is defined as a tuple $(\mathcal{S}_v, \mathcal{S}_h, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{C}, b_0)$, where $\mathcal{S}_v$ is the space of fully observable states; $\mathcal{S}_h$ is the space of hidden continuous states; $\mathcal{A}$ is the set of actions; Ω is the set of observations; $\mathcal{T}$ is the state transition function; $\mathcal{O}$ is the observation function, such as $\mathcal{O}(o, a, s'_h, s'_v) = \Pr(o|s'_h, s'_v, a)$; $\mathcal{C} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_0^+$ is the cost function; $b_0 = (s_v^0, b_{\mathcal{S}_h}^0)$, where $b_{\mathcal{S}_h}^0 \in \mathcal{B}_h$ is the initial probability distribution over the initial hidden continuous states, conditioned to $s_v^0 \in \mathcal{S}_v$, the initial fully observable state.

The visible state $s_v \in \mathcal{S}_v$ is defined as a tuple $s_v = (F_{S_1}, F_{S_2}, ..., F_{S_N}, F_{Col}, \mathbf{P})$, where F_{S_i} defines the fully observable boolean state variable for the availability of sensor S_i, $i = 1, ..., N$, and F_{Col} represents a fully observable boolean variable for a collision flag. Finally, $\mathbf{P}$ is the localization error covariance matrix computed by the Kalman filter.

The hidden continuous state $s_h \in \mathcal{S}_h$ is defined as $s_h = \mathbf{x}$, recalling that $\mathbf{x}$ is the continuous vehicle state vector defined by $\mathbf{x} = \begin{bmatrix} \mathcal{X}^T & \mathcal{V}^T & \beta_a^T \end{bmatrix}^T$. Apart from the vehicle position $\mathcal{X}$, it is necessary to consider the velocity $\mathcal{V}$ and the accelerometer bias β_a in the state s_h, as they are considered in the transition function to estimate the next state.

An action $a \in \mathcal{A}$ is defined as a tuple $a = (\{\mathcal{V}_{ref}\}, m_n)$, where $\mathcal{V}_{ref}$ is the reference velocity in the guidance module, $\{\mathcal{V}_{ref}\}$ defines a finite set of possible $\mathcal{V}_{ref}$ and $m_n \in \{S_1, ..., S_N\}$ is the navigation mode to be considered at each planning epoch t, depending on the sensors' availability.

Given the specificity of the model addressed, the set of observations Ω is equal to $\mathcal{S}_v$. Although the agent receives no direct or even imprecise observation on the state s_h, the propagation dynamics of the error execution depending on the choice of navigation mode are known. This avoids considering sensor measurements, which are not accessible at the moment of planning, and therefore avoid working with a continuous observation space.

Following the GNC transition model described in [3], the complete transition function is given as:

$$\mathcal{T}(s_v, s'_v, a, s'_h, s_h) = \Pr(s'_v|s'_h) \Pr(s'_h|s_h, s_v, a) \quad (1)$$

where $\Pr(s'_v|s'_h)$ represents the transition function for s'_v, which depends only on the probabilistic sensors availability maps and $\Pr(s'_h|s_h, s_v, a) \sim \mathcal{N}(\bar{s}'_h, \tilde{\Sigma}'(s_v))$ is based on the GNC closed-loop model, given that the probability distribution of a predicted state s'_h follows a normal distribution $\mathcal{N}(\bar{s}'_h, \tilde{\Sigma}'(s_v))$, which in turn, is a function of the previous state s_h, the previous visible state s_v and the action a.

The cost function to be minimized is defined in Equation 2. It is expected that by minimizing the cost, the algorithm will minimize the flight time (for efficiency) and the probability of collision (for safety) at the same time.

$$\mathcal{C}(s_t, a_t) = \begin{cases} 0, & \text{if } s_t \in \mathcal{G} \\ K - \sum_{k=0}^{t-1} \mathcal{C}(s_k, a_k), & \text{if } s_t \text{ in collision} \\ f_t, & \text{otherwise} \end{cases} \quad (2)$$

where f_t is the flight time for a given action a at decision step t and K is a fixed cost in case of collision. When a collision occurs, the cost of any action is a fixed penalty subtracted with the total flight time since the initial belief state until the collision state. This trick avoids penalizing more if the collision occurs after a longer flight time or near the goal.

The aim of solving a POMDP problem is to find a policy $\pi : \mathcal{B} \rightarrow \mathcal{A}$, where $\mathcal{B}$ define the belief state space, which optimizes a given criterion usually defined by a value function. In the MOMDP problem addressed, the value function $V^\pi(b)$ is defined as the expected total cost payed from starting in a certain belief state b and following policy π, as:

$$V^\pi(b) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \mathcal{C}(b_t, \pi(b_t)) \middle| b_0 = b \right] \quad (3)$$

where $\mathcal{C}(b_t, \pi(b_t) = a)$ is the expected cost of executing an action a in belief state b. Then the optimal value function $V^*(b)$ is given as:

$$V^*(b) = \min_{a \in \mathcal{A}} \left[\mathcal{C}(b, a) + \sum_{s_v \in \mathcal{S}_v} \Pr(s_v|b, a) V^*(b_a^{s_v}) \right] \quad (4)$$

When Equation 4 converges for all belief states, it is possible to extract the optimal policy π^*:

$$\pi^*(b) = \arg\min_{a \in \mathcal{A}} \left[\mathcal{C}(b, a) + \sum_{s_v \in \mathcal{S}_v} \Pr(s_v|b, a) V^*(b_a^{s_v}) \right] \quad (5)$$

3. Offline Approach

3.1. POMCP-GO Algorithm

Given the specificity of the MOMDP model chosen, comprising a continuous hidden state space and a discrete fully observable state space, the maintenance and update of belief states would require an expensive computational effort. An interesting solution is to apply algorithms that do not need explicit representations of belief states at each decision stage, such as the POMCP algorithm [12].

Nevertheless, the classical POMCP algorithm does not completely meet the needs for path planning towards a goal state, therefore a new variant was proposed. Such algorithm is called POMCP-GO (Algorithm 1) as it is a goal oriented version of the POMCP algorithm. The main differences between both versions are hereafter discussed.

Algorithm 1: POMCP-GO Offline

```
1  Function POMCP-GO Offline($h, b_0$):
2      while $nbTrial < nb_{max}$ do
3          $s_h \sim b_0$
4          Trial($h$, $s_h$, $s_v$, 0)
5          $nbTrial += 1$
6      return $a^* \leftarrow \arg\min_{a \in \mathcal{A}} Q(b_0, a)$
7  Function Trial($h$, $s_h$, $s_v$, $d$):
8      if $s_h \in \mathcal{G}$ then
9          return 0
10     if $F_{Col} == 1$ then
11         return $K - t\, f_t$
12     if $h \notin T$ then
13         for $a \in \mathcal{A}$ do
14             $T(ha) \leftarrow (N_{init}(ha), Q_{init}(h,a))$
15         $V_{init}(h) \leftarrow \min_{a \in \mathcal{A}} Q(h,a)$
16     $\bar{a} \leftarrow \arg\min_{a \in \mathcal{A}} \left\{ Q(h,a) - c\sqrt{\frac{\log N(h)}{N(ha)}} \right\}$
17     $(s'_h, s'_v, \mathcal{C}(s_h, \bar{a})) \sim G(s_h, \bar{a})$
18     $Q(h,\bar{a})' \leftarrow \mathcal{C}(s_h, \bar{a})$ + Trial($hao$, $s'_h$, $s'_v$,
           $d+1$)
19     $N(h) \leftarrow N(h) + 1$
20     $N(h\bar{a}) \leftarrow N(h\bar{a}) + 1$
21     $Q(h,\bar{a}) \leftarrow Q(h,\bar{a}) + \frac{Q(h,\bar{a})' - Q(h,\bar{a})}{N(ha)}$
22     $V(h) \leftarrow \min_{a \in \mathcal{A}} Q(h,a)$
```

Firstly, a computational budget as a number of trials to perform is used (line 2). In an offline configuration, all trials start from the initial belief state $V(b_0)$, that is, the root of the exploration tree (line 3), rather than at the current belief state. Then, belief states are represented indirectly through the POMCP property of generating sequences of history nodes and sequence nodes. Since it is a goal-oriented problem, [3] proposed that each trial should end up only in a terminal state (goal or collision), instead of being stopped every time a new node is initialized. In the case of reaching a goal, a cost of 0 is returned (line 9); whereas if collision is detected, a collision cost is returned (line 11), as defined by Equation 2.

Differently from POMCP [12], in POMCP-GO, if a new node ha is created, its value is estimated using an initial heuristic value and not a rollout method. Indeed, Q-values are initialized using the Dijkstra deterministic shortest path algorithm [5] applied to the obstacles grid map to compute the estimated flight time. Furthermore, as the Q-value is progressively updated throughout the simulations, it allows to compute the value of $V(h)$ as $V(h) = \min_{a \in \mathcal{A}} Q(h,a)$ (line 15). At each step, the action a is chosen with a UCB1-based selection strategy (line 16), the next new state s'_h is calculated using the GNC transition function, rather than the black-box simulator, and the visible state s'_v is randomly drawn according to the probabilities given by the sensor's availability maps. The algorithm then continues to explore from the new node $hao = \{h, a, s'_v\}$.

3.2. Action selection strategies for planning

In our application case, POMCP-GO relies on the following UCB1-based formula for action selection:

$$\bar{a}_{\text{UCB}} = \arg\min_{a \in \mathcal{A}} \left\{ Q(h,a) - c\sqrt{\frac{\log N(h)}{N(h,a)}} \right\} \qquad (6)$$

Equation 6 depends on an exploration factor c, which is a constant value typically adjusted manually. This parameter varies significantly between planning domains, requiring an exhaustive search to find the most suitable value. Since this intensive parameter tuning is not suited for an online planning configuration, the performance of three other selection strategies is examined: decay with depth (DWD), entropy-based coefficient (EBC) and a two-steps sampling scheme based on simple regret and cumulative regret. While the first two techniques explore the use of adaptive coefficients in Equation 6 to avoid the exhaustive search, the last approach introduces the notion of simple regret minimization into the selection procedure.

3.2.1. Decay with depth (DWD)

In online path planning, it is often more interesting to explore at the beginning or near obstacles than at the end, when the vehicle is close to the goal. Indeed, it is during the first periods that it is crucial to explore alternative paths, which anticipate likely collisions. Using the formula proposed by [3], the DWD coefficient starts with a higher value in the beginning, to support the exploration of available actions, and then as the depth of the tree increases, lower values of the coefficient are considered in order to give way to the exploitation, as follows:

$$c_{DWD} = \frac{C_k}{t}\,(K - t\, f_t) \qquad (7)$$

where C_k is a constant value, K is the collision penalty and t is the depth of the tree since the beginning of the flight, i.e. $t = t_0$.

3.2.2. Entropy-based Coefficient (EBC)

The suggested coefficient is based on a score related to a measure of the uncertainty about the sensors' availability. More specifically in our application case, the coefficient adapts according to the entropy of the probability grid map of the navigation sensors' availability. Then, during the planning process, the value of the coefficient lies within a specific user-defined interval $[c_{min}, c_{max}]$, encoding, on the action selection strategy, the need to explore more thoroughly in the areas where the uncertainty is higher. The EBC is proportional to the maximum possible cost value as follows:

$$c_{EBC}(s_v, s_h) = e_n(s_v, s_h) \max_{\substack{s \in \mathcal{S} \\ a \in \mathcal{A}}} |\mathcal{C}(s, a)| \tag{8}$$

where $e_n(s_v, s_h) = (c_{max} - c_{min})\, e(s_v, s_h) + c_{min}$ is the normalized entropy to fit in $[c_{min}, c_{max}]$, and

$$e(s_v, s_h) = -\sum_{s_v \in \int_v} \Pr(s_v|s_h) \log_2(\Pr(s_v|s_h)) \tag{9}$$

is the value of the entropy according to the probability grid map of the navigation sensors' availability. In this case, $\int_v = (F_{S_1}, F_{S_2}, ..., F_{S_N})$ is a subset of $\mathcal{S}_v$ that only considers the fully observable boolean state variables F_{S_i}, i.e. neither $\mathbf{P}$ nor F_{Col} are accounted for during the computation of this coefficient.

3.2.3. SR+CR MCTS Sampling Scheme

In [13], authors argue that, at the root node, the sampling in MCTS is usually aimed at finding the first move to perform. For this, one needs to promote exploration. Once one move is shown to be the best choice with high confidence, the value of information of additional samples of the best move is low. On the other hand, it is beneficial to promote exploitation in the nodes deeper in the tree. For this purpose, [13] propose a two-stage sampling scheme, SR+CR MCTS. This selects an action at the current root node according to a scheme suitable for minimizing the simple regret (SR), which can be achieved by replacing the $\log(\cdot)$ operator in Equation 6, with a faster growing sub-linear function, such as $\sqrt{(\cdot)}$:

$$\bar{a}_{\text{UCB}_{\sqrt{(\cdot)}}} = \arg\min_{a \in \mathcal{A}} \left\{ Q(h, a) - c\sqrt{\frac{\sqrt{N(h)}}{N(h, a)}} \right\} \tag{10}$$

And then at non-root nodes it selects actions according to UCB1, which approximately minimizes the cumulative regret (CR). In this case, the exploration factor c in Equation 10 has to be tuned in the same manner as in UCB1. However, the authors argue that this approach is expected to be significantly less sensitive to the tuning of this parameter and also to outperform the classic UCB1 in terms of convergence rate of the value function.

3.3. Back propagation strategies

The backup function defines how the knowledge on state-value estimates $V(h)$ and action-value estimates $Q(h, a)$ that is gathered in the trials is propagated through the tree [6]. In this work, two different Q-value approximations are tested, resulting in two distinct back-propagation strategies. For both approaches, the state-value estimates are updated based on the best successor.

3.3.1. Classical POMCP

In this strategy, the action-value estimates are updated based on the mean return from all trajectories started when action a was selected in history h. Let $Q(h, a)'$ be the simulated return computed recursively over the trajectories, then the state-value and action-value estimates are calculated as

$$\begin{cases} V(h) \leftarrow \min_{a \in \mathcal{A}} Q(h, a) \\ Q(h, a) \leftarrow Q(h, a) + \dfrac{Q(h, a)' - Q(h, a)}{N(ha)} \end{cases} \tag{11}$$

This strategy is the one commonly adopted in the POMCP algorithm. As the action-value estimate $Q(h, a)$ averages over all trajectories started in that action a, and not over trials starting with a and following the optimal policy, this might cause a potential pitfall: if a trajectory yields a very high cost compared to an optimal one, a single trial over said course can bias $Q(h, a)$ disproportionately over many trials [6].

3.3.2. MinPOMCP

This strategy derives from the MaxUCT algorithm proposed in [6] and constitutes part of the contributions of this dissertation. In this case, the estimation of the action-values is based on the value of its best successor, rather than on all trajectories, as follows

$$\begin{cases} V(h) \leftarrow \min_{a \in \mathcal{A}} Q(h, a) \\ Q(h, a) \leftarrow C(h, a) + \dfrac{\sum_{hao} N(hao)\, V(hao)}{N(ha)} \end{cases} \tag{12}$$

where

$$C(h, a) \leftarrow C(h, a) + \frac{\mathcal{C}(s_h, a) - C(h, a)}{N(ha)} \tag{13}$$

is the mean immediate cost of executing action a in history h. As a result of applying this strategy, the contributing subtree in the back-propagation step is identical to the best partial solution tree and, therefore, the pitfall discussed in the Classical POMCP strategy no longer applies.

3.4. Simulations

Configuration A total of 8 case studies are examined: 2 environment maps, for which 2 distinct sensor's availability maps, with 2 different initial states each, are considered. A total of 10 value and policy optimizations are performed for each case study. The total number of trials in each optimization process is set to 50000. For each 5000 trials, 1000 simulations are performed to evaluate the policy being currently optimized.

Two onboard sensors are considered: INS, available all the time, and GPS, available according to the probability maps. Moreover, two benchmarks from [8] are selected: *WallBaffle*: map containing two walls as obstacles (Figure 2a) and *CubeBaffle*: map containing two cubes as obstacles (Figure 2d). Both maps contain a grid size of $100 \times 100 \times 20$ cells, where each grid cell has the size of $2m \times 2m \times 2m$.

The initial belief state is defined as $b_0 = (s_v^0, b_{\mathcal{S}_h^0} = (\bar{s}_h^0, \tilde{\Sigma}_0))$, where $s_v^0 = [1, 1, 0, \mathbf{P}]$ is the initial visible state; $\mathbf{P} = \tilde{\Sigma}_0 = diag(1, 1, 1, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01)$. Two initial states s_h^0 are considered for each map. For *WallBaffle*: $s_h^0 = [10, 25, 5, 0, 0, 0, 0, 0, 0]$ and $s_h^0 = [50, 25, 5, 0, 0, 0, 0, 0, 0]$, representing the initial positions $(10, 25, 5)m$ and $(50, 25, 5)m$, respectively; and for *CubeBaffle*: $s_h^0 = [35, 20, 5, 0, 0, 0, 0, 0, 0]$ and $s_h^0 = [65, 20, 5, 0, 0, 0, 0, 0, 0]$, representing the initial positions $(35, 20, 5)m$ and $(65, 20, 5)m$, respectively. The target state is defined at the position $s_g = (50, 80, 5)m$.

The set of $\mathcal{V}_{ref}$ is composed by 26 reference speeds for each navigation sensor, thus comprising a total of 52 possible actions. Parameters in the GNC model are configured to ensure an action duration of 4 seconds. The collision cost is fixed at $K = 450$.

Concerning the selection strategies, extensive search is performed in UCB1 (UCB(c)) for the values $c = (0.01, 0.05, 0.1, 0.5, 1, 5, 10)$, Decay with Depth (DWD) uses $C_k = 0.2222$, Entropy-based Coefficient (EBC) applies $[c_{min}, c_{max}] = [0, 0.0222]$, so that c_{EBC} lies within an interval of $[0, 10]$, to be comparable with the values examined in UCB1, and SR+CR MTCS ($\text{UCB}_{\sqrt{(\cdot)}}(c)$) only uses the coefficient that showed the best results in UCB1.

The metrics evaluated include the *value of the initial belief state $V(b_0)$ during the optimization* (used to check the convergence in value of the policy being optimized for the initial state of belief), *value of the initial belief state $V(b_0)$ executed* (based on the simulations), the *success rate* (in %), the *average flight time T* for the successful simulations (allows to verify the minimization of the flight time) and the *computational time per optimization* (indicator of the computational complexity inherent to the strategies).

Classical POMCP is tested for each of the 4 selection strategies, while MinPOMCP is performed only for the coefficient that showed the best results in UCB1 and for the EBC, making it a total of 6 different combinations used. Selection strategies using the MinPOMCP back-propagation approach are identified with an extra subscript MP.

Results This section comprises only the values obtained after the 50000 trials were reached. Figures 3 and 4 represent the results of the metrics $V(b_0)$ optimized and $V(b_0)$ simulated for all the approaches. The data is organized in the form of *bars*, being the tip of the filled bar the average, while the extremes of the vertical lines represent the standard deviation.

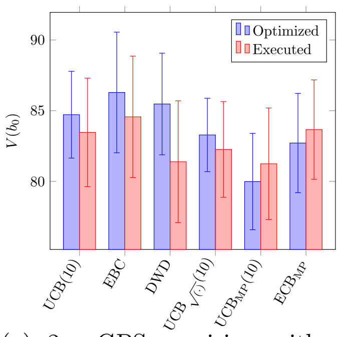
It is noticeable that the UCB1's best fixed coefficient varies significantly, not only across the different GPS precision probabilistic availability maps considered, but also when changing initial positions in the same map. For this reason, the use of an adaptive coefficient to fit all the different planning domains becomes a very promising solution for an online planning configuration. Both $V(b_0)$ optimized and simulated computed for EBC lie within the values obtained for the UCB1 fixed coefficients for every case study, although never reaching the values of the best fixed coefficient. Therefore, a compromise must be made between the desired quality of the results and the time one is willing to spend on the exhaustive search for the optimal coefficient value. On the other hand, both DWD and $\text{UCB}_{\sqrt{(\cdot)}}$ do not show a consistent behaviour, yielding the best results for particular case studies, while the worst results for others. Furthermore, the DWD strategy takes more time to ensure the convergence of the value function. Finally, the combination of ECB with the MinPOMCP strategy is a promising approach for the online planning, since it offers no need for extensive search of the best UCB1 coefficient and the MinPOMCP helps to accelerate value convergence.

4. Online Approach

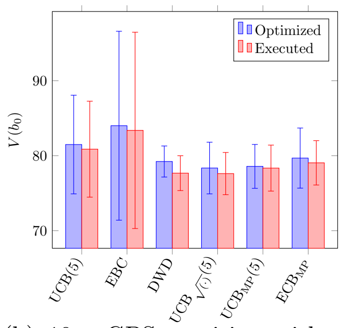
4.1. Online POMCP-GO algorithm

The main differences between the online and the offline approach include the computational budget, the belief state representation and the trial length.

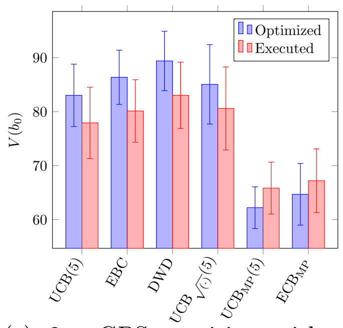
Regarding the computational budget, a timeout is applied, instead of a fixed number of trials. Being an online approach, the trials begin from the current belief state b, rather than starting always in the initial belief. As such, rejection sampling [12] is performed to reinvigorate the new belief state, after each action a is executed and the respective observation s_v is perceived from the real world. Similarly to the classic POMCP [12], each trial finishes after a new node is created. Furthermore, an additional trial length condition is established, ending the trial if the current depth reached a maximum depth D. This condition promotes breadth-first search, which improves action selection with short time budgets.



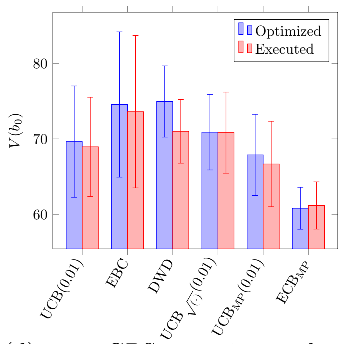
(a) $2m$ GPS precision with initial position at $(10, 25, 5)m$



(b) $10m$ GPS precision with initial position at $(10, 25, 5)m$

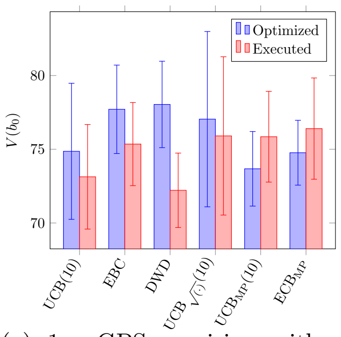


(c) $2m$ GPS precision with initial position at $(50, 25, 5)m$

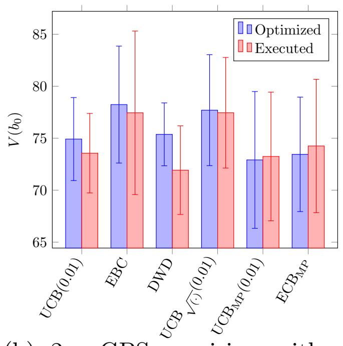


(d) $10m$ GPS precision with initial position at $(50, 25, 5)m$

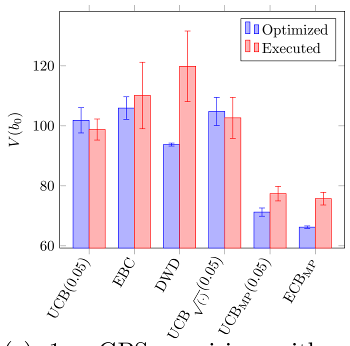
Figure 3: Results obtained for the *WallBaffle* map, using different combinations of action selection and back-propagation strategies.



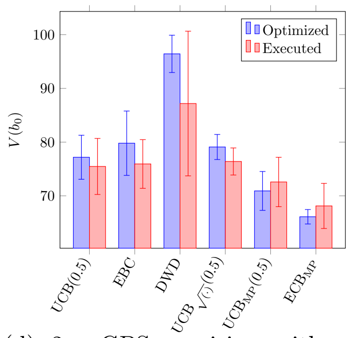
(a) $1m$ GPS precision with initial position at $(35, 20, 5)m$



(b) $2m$ GPS precision with initial position at $(35, 20, 5)m$



(c) $1m$ GPS precision with initial position at $(65, 20, 5)m$



(d) $2m$ GPS precision with initial position at $(65, 20, 5)m$

Figure 4: Results obtained for the *CubeBaffle* map, using different combinations of action selection and back-propagation strategies.

This online approach acts by interleaving planning and execution phases. Because the ultimate goal of these guidance and navigation strategies is to apply them onboard the UAV in a real-time configuration, it becomes unfeasible to have the UAV constantly stopping mid-air while the planning phase takes place. For this reason, the POMCP-GO is incorporated in a planning-while-executing framework described in the next section.

4.2. Planning-while-executing framework

AMPLE [11] is a framework that follows the direction of reactive or continuous planning, in which a plan is initially proposed and then works by iteratively fixing flaws in such plan when an anomaly occurs at execution-time, and on continuously updating goals, state and planning horizons.

This framework is *strictly anytime* in the sense of policy execution under time constraints, as it ensures the return of an applicable action in any possible execution state at a precise time point, exactly when required by the execution engine; *reactive* to environment changes, which are incorporated in the planning requests being managed in parallel to the action execution and *conditional* by prioritizing future execution states and computing a partial (incomplete) but applicable policy for each action.

The AMPLE framework, depicted in Figure 5 is composed of two threads: a *planning thread* (server), that manages plan optimization while answering to client requests; and an *execution thread* (client), that adds and removes planning requests according to the system's and environment's evolution, in order to get the action to execute in the current state, from the optimized policy or from the default one.

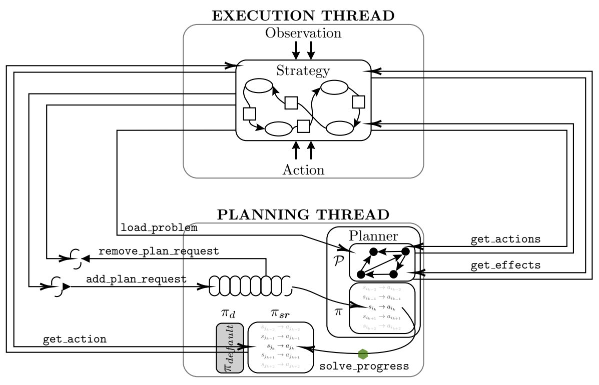


Figure 5: AMPLE architecture: connections between the execution and the planning threads [11].

4.3. Integration of POMCP-GO in AMPLE

The AMPLE framework is developed in the Robotic Operating System (ROS). As such, three nodes are established for the planning thread, the execution thread (using the `AMPLE-NEXT` strategy [11]) and the planner (online POMCP-GO). The several methods illustrated in Figure 5 constitute services that connect these nodes. The planning thread node is implemented as an intermediary in all of these services, meaning that the execution thread node and the planner node are not directly connected.

In this application case, the planning time for each future state s'_v is computed proportionally to the probability of getting such effect after the execution of an action a in the current belief state b, i.e. $\Pr(s'_v|b,a)$. However, the MOMDP model is not declarative on these probabilities, so an approximation is done with the following formula:

$$\Pr(s'_v|b,a) = \frac{N(hao)}{N(ha)} \tag{14}$$

As the values of the visitation counters $N(hao)$ and $N(ha)$ are iteratively updated in the back-propagation step of the POMCP-GO algorithm during the trials, this probability becomes more precise as the mission progresses.

4.4. Simulations

Configuration Simulations are conducted comparing the performance of three approaches: POMCP-GO driven in AMPLE; POMCP-GO implemented using the classical online paradigm, that is, with planning and execution phases interleaved; and an algorithm that consecutively applies default actions with no planning time associated. The latter serves as a baseline indicator of the quality of the default policy π_d used. There are a total of 4 application cases examined: 2 environment maps, using 1 sensor's availability map with 2 different initial belief states. A total of 100 simulations are performed for each application case.

Table 1 summarises the parameters used in these simulations. The metrics evaluated include the *total mission duration*, from the moment the problem is loaded until a terminal state is reached (either a goal or a collision); the *total planning time* during the mission, taking into consideration the initial bootstrap time in the case of AMPLE; the *success rate* (in %) and the *default actions rate*, which is the ratio between the number of default actions, computed from the default policy π_d, used and the total number of actions performed (in %).

	POMCP-GO	POMCP-GO in AMPLE
Paradigm	Planning and execution phases interleaved	Concurrent planning and execution phases
timeout	$2s$, $3s$, $4s$	$4s$ (action duration)
$\Delta_{bootstrap}$	–	$5s$, $10s$, $15s$
ASS	UCB1* and EBC	UCB1* and EBC
BPS	Classical POMCP	Classical POMCP
D	10	10
K	450	450
#Particles b	300	300

Table 1: Parameters used for the online simulations, where ASS stands for Action Selection Strategy and BPS for Back-propagation Strategy.

Results Figure 6 depicts the results obtained for the *WallBaffle* map, while Figure 7 illustrates the results obtained for the *CubeBaffle* map, using the EBC as an action selection strategy.

As expected, increasing the time budget in the POMCP-GO algorithm leads to better success rates, since providing more time for planning allows the algorithm to search deeper in the tree and thus to improve the quality of the policy being optimized. On the other hand, such 2 extra seconds in the *timeout* increases significantly the total duration of the mission, by more than 35% in most cases.

Similarly in the AMPLE framework, increasing the initial bootstrap time improves the performance of the algorithm. This happens because providing more time to the planner to optimize the action for the first belief state also allows it to explore more thoroughly the state space near the root, improving the policy for the initial belief states. However, the success rates achieved by AMPLE are in general worse than when using POMCP-GO in the classical online paradigm, except when comparing POMCP-GO with a timeout of $2s$ and AMPLE with bootstrap of $\Delta_{bootstrap} = 15s$. This stems from three factors. Firstly, in the POMCP-GO the time budget is used entirely to compute the next optimized action, without limiting the tree to search in a specific subtree. On the other hand, in the AMPLE framework, the planning time given is proportional to $\Pr(s'_v|b,a)$ (Equation 14), and consequently during such duration the tree is limited to search in the subtree correspondent to that effect. Furthermore, since the algorithm has to learn these probabilities as the mission progresses, it becomes fairly possible for the planner to assign the lowest planning time to the effect that was posteriorly perceived from the environment.

Secondly, the POMCP-GO is an online algorithm with anytime properties. However, it does not account for its own computation time during the optimization process and, consequently, if the *timeout* is reached before the end of a trial, the algorithm will only provide an action after the back-propagation step of said trial is performed, allowing to further improve the quality of the policy. Whereas in the AMPLE framework, the algorithm is strictly anytime, meaning it does not wait for the end of the trial and automatically provides the action already present in the backup policy $\pi_{\mathbf{sr}}$.

Thirdly, the AMPLE framework runs in ROS, having three distinct nodes connected through services. This causes the planning requests to naturally inherent a lag, meaning the actual planning time budget given to an effect is lower than the one included in the planning request. This can have as a consequence that no actual planning is done for that effect, leading to a default action being used instead.

Apart from the loss of optimality found in AMPLE, this approach allows the reduction of the overall time of the mission, exploiting action execution time to anticipate and plan future states.

5. Application in Gazebo

To test the AMPLE framework with the POMCP-GO algorithm in Gazebo, the hector quadrotor package [9] is used in an environment that replicates the obstacle maps.

Figure 8 represents five trajectories planned for the *WallBaffle* map, using a probability map of $2\,meter$ GPS precision for an initial position in $(50, 25, 5)m$. A bootstrap duration of $\Delta_{bootstrap} = 15s$ is used for AMPLE, as well as the EBC in action selection and Classical POMCP as a back-propagation strategy. Projections of the trajectories are illustrated over the 3 planes xy, xz and yz.

The UAV chooses to go between the obstacles in order to minimize the flight time, instead of flying over them. This choice leads the UAV through areas where the availability of the GPS is less likely, causing the navigation solution chosen to be predominantly INS. This has an impact on the width of the simulated trajectory, which reflects the execution error inherent to the INS.

6. Conclusions

The first contribution of this dissertation was a parameter tuning stage for the planning algorithm. Comparisons between the proposed approaches allowed to conclude that while DWD and UCB$_{\sqrt{\cdot}}$ did not show a consistent behaviour, the EBC results always lied within the values obtained for the fixed coefficients in UCB1, although never reaching the values of the best fixed coefficient. Therefore, a compromise must be made between the desired quality of the results and the time one is willing to spend on the exhaustive search for the optimal value as done in UCB1. Furthermore, the back-propagation MinPOMCP generally improved the performance for both UCB1 and EBC selection strategies, accelerating the convergence of the value function as expected, following the work of [6].

Simulation results regarding the combined approach of POMCP-GO and AMPLE, showed that although AMPLE did not achieve in general success rates as good as the POMCP-GO algorithm, the overall time of the mission was reduced and actions were ensured to be provided strictly anytime.

As future work, we propose to test the combination of AMPLE and POMCP-GO with the MinPOMCP back-propagation strategy, which is expected to achieve better results than the ones obtained in Section 4.4 and also to test it in more complex and realistic scenarios with different dynamics that truly reflect an urban area. Furthermore, we propose to explore a parallelization of planning requests in the AMPLE framework, rather than the current sequentialization strategy being used, to provide the same planning duration for each request.

References

[1] M. W. Achtelik and S. Lynen. Motion- and Uncertainty-aware Path Planning for Micro Aerial Vehicles. *Journal of Field Robotics*, 31(4):676–687, 2014.

[2] C. P. Chanel, F. Teichteil-Königsbuch, and C. Lesire. Multi-target detection and recognition by UAVs using online POMDPs. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013*, pages 1381–1387, 2013.

[3] J.-A. Delamer, Y. Watanabe, and C. P. Carvalho Chanel. Solving path planning problems in urban environments based on a priori sensors availability and execution error propagation. In *AIAA Scitech 2019 Forum*, 2019.

[4] J.-A. Delamer, Y. Watanabe, and C. Ponzoni Carvalho Chanel. MOMDP modeling for UAV safe path planning in an urban environment. In *Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA 2017)*, 2017.

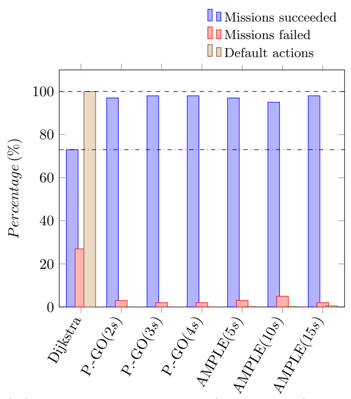
[5] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematlk*, 1(1):269–271, 1959.

[6] T. Keller and M. Helmert. Trial-based Heuristic Tree Search for Finite Horizon MDPs. In *Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS'13)*, pages 135–143, 2013.

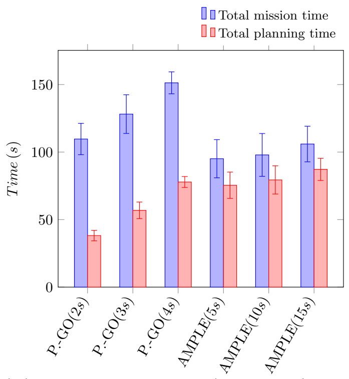
[7] F. Kleijer, D. Odijk, and E. Verbree. Prediction of GNSS Availability and Accuracy in Urban Environments Case Study Schiphol Airport. In *Location Based Services and TeleCartography II*, Lecture Notes in Geoinformation and Cartography, pages 387–406. Springer, Berlin, Heidelberg, 2009.

[8] B. Mettler, Z. Kong, C. Goerzen, and M. Whalley. Benchmarking of obstacle field navigation algorithms for autonomous helicopters. *Annual Forum Proceedings - AHS International*, 3:1936–1953, 2010.

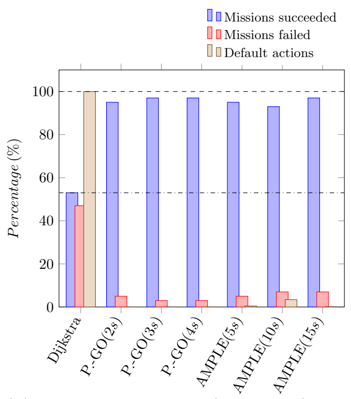
[9] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. von Stryk. Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo. In *Third International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 400–411. Springer, Berlin, Heidelberg, 2012.



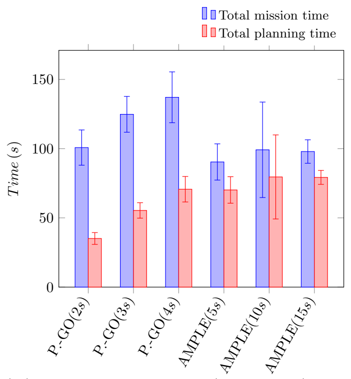
(a) Initial position $(10, 25, 5)m$



(b) Initial position $(10, 25, 5)m$

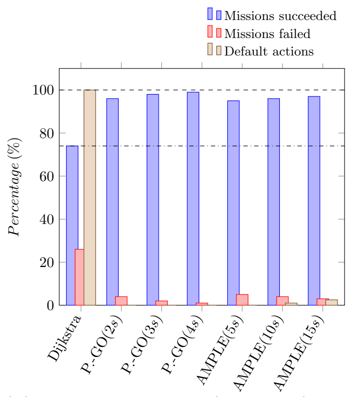


(c) Initial position $(50, 25, 5)m$

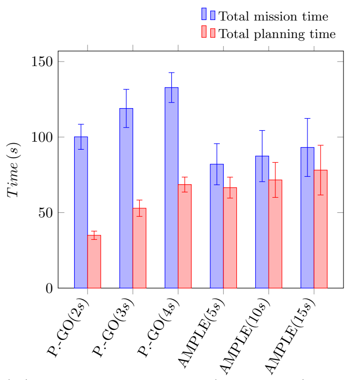


(d) Initial position $(50, 25, 5)m$

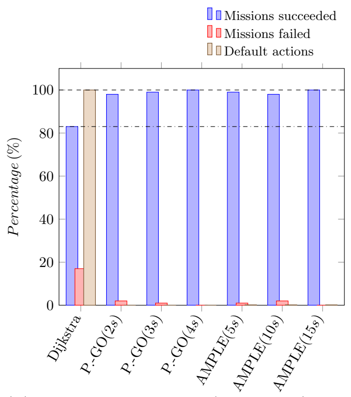
Figure 6: Results obtained for the *WallBaffle* map, using a probability grid map considering $2m$ GPS precision, the EBC as action selection strategy and Classical POMCP as back-propagation strategy.



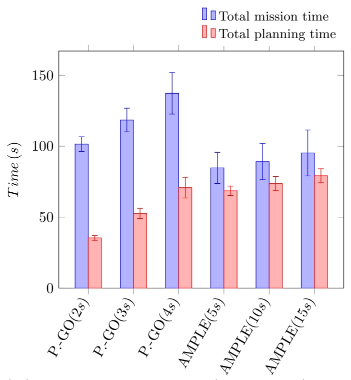
(a) Initial position $(35, 20, 5)m$



(b) Initial position $(35, 20, 5)m$



(c) Initial position $(65, 20, 5)m$



(d) Initial position $(65, 20, 5)m$

Figure 7: Results obtained for the *CubeBaffle* map, using a probability grid map considering $2m$ GPS precision, the EBC as action selection strategy and Classical POMCP as back-propagation strategy.

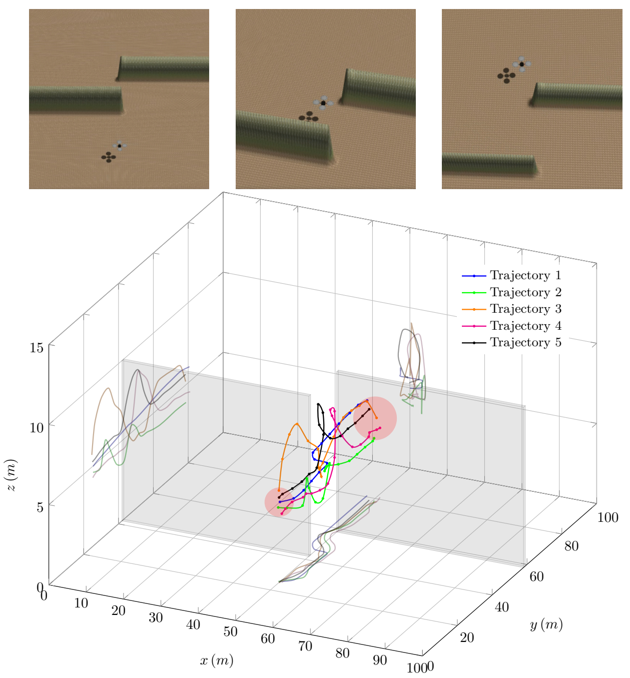


Figure 8: Five trajectories executed in the *WallBaffle* environment in Gazebo, for an initial position at $(50, 25, 5)m$ and a goal position at $(50, 80, 5)m$.

[10] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning under Uncertainty for Robotic Tasks with Mixed Observability. *International Journal of Robotics Research*, 29(8):1053–1068, 2010.

[11] C. Ponzoni Carvalho Chanel, A. Albore, J. T'Hooft, C. Lesire, and F. Teichteil-Königsbuch. AMPLE: an anytime planning and execution framework for dynamic and uncertain problems in robotics. *Autonomous Robots*, 43(1):37–62, 2019.

[12] D. Silver and J. Veness. Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems 23 (NIPS 2010)*, pages 2164–2172, 2010.

[13] D. Tolpin and S. E. Shimony. MCTS Based on Simple Regret. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[14] F. Vanegas, D. Campbell, N. Roy, K. J. Gaston, and F. Gonzalez. UAV tracking and following a ground target under motion and localisation uncertainty. In *2017 IEEE Aerospace Conference*, pages 1–10. IEEE, 2017.

[15] Y. Watanabe, S. Dessus, and P. Fabiani. Safe path planning with localization uncertainty for urban operation of VTOL UAV. In *AHS 70th Annual Forum*, 2014.