MOMDP solving algorithms comparison for safe path planning problems in urban environments

Jean-Alexis Delamer[1] and Yoko Watanabe[2] and Caroline P. Carvalho Chanel[3]

***Abstract*— This paper tackles a problem of UAV safe path planning in an urban environment where the onboard sensors can be unavailable such as GPS occlusion. The key idea is to perform UAV path planning along with its navigation an guidance mode planning where each of these modes uses different set of sensors and whose availability and performance are environment-dependent. It is supposed to have a-priori knowledge in a form of gaussians mixture maps of obstacles and sensors availabilities. These maps allow the use of an Extended Kalman Filter (EKF) to have an accurate state estimate. This paper proposes a planner model based on Mixed Observability Markov Decision Process (MOMDP) and EKF. It allows the planner to propagate such probability map information to the future path for choosing the best action minimizing the expected cost.**

I. Introduction

Safe navigation of autonomous vehicles in urban environment is a challenging problem. These vehicles rely on their onboard sensors to navigate through the environment. Their navigation performance depends directly on the onboard sensors whose availability and precision can vary with the environment. For example, the GPS localization precision depends on the satellites constellation and their visibilities. It is, however, possible to predict its localization precision, called Dilution of Precision (DOP), for a given environment [12]. Such information can be used as a priori knowledge in the path planning task, to ensure the safety under uncertainty.

In this context, this paper tackles such safe path planning problem for autonomous vehicles in urban environments, in particular for UAVs (Unmanned Aerial Vehicles). [18] and [1] have addressed UAV path planning problems, by considering the localization uncertainty which is propagated along a planned path in function of its environment. For instance, [18] applies the A* algorithm and makes use of uncertainty corridor to evaluate the plan for choosing the most efficient and safe path. [7] and [1] propagate the position uncertainty during path search by using RRBT algorithm. However, any of these approaches consider the complete GNC (Guidance, Navigation, and Control) closed-loop vehicle kinematics model into the decisional process.

The UAV safe path planning problem, addressed in this paper, is modeled as a particular Mixed-Observability Markov Decision Process (MOMDP) [16]. MOMDP is an extension of the classical Partially Observable Markov Decision Process (POMDP) [11], that allows the factorization of the state variables into fully and partially observable state variables. It holds in a smaller belief state space dimension, accelerating policy computation. The transition and observation functions of the MOMDP are built on the vehicle GNC model, and on the a priori knowledge of the environment given as probability grid maps of obstacles or sensor availabilities, respectively. Through these complex functions which combine continuous state variables transitions with discrete grid maps for sensor availability observations, the resulting updated belief state has a particular form. To address this difficulty, the belief state is approximated by a Gaussian Mixture Model (GMM) using the Expectation-Minimization (EM) algorithm [2].

Moreover, another particularity of this planning problem arises with the cost function proposed in this paper. It holds in having a non piecewise linear and convex (non-PWLC) value function [11], which prevents from using classical MOMDP solvers [16], [3]. Consequently, this paper presents two algorithms which do not require the PWLC property. The first algorithm is based on (L)RTDP (Labelled Real-Time Dynamic Programming) [5] and RTDP-bel [6] algorithms. (L)RTDP [5] improves the convergence of RTDP (and RTDP-bel in consequence) by labeling the already converged states. RTDP-bel use an hash-table only defined for belief states visited during policy learning. The second algorithm is based on POMCP [17], a Monte-Carlo tree search algorithm for partially observable environments. POMCP, as UCT (Upper Confidence bounds applied to Trees) [13], applies the UCB1 (Upper Confidence Bounds) greedy action selection strategy. POMCP approaches the value of a belief state by the average of evaluated costs during simulations which have started from this belief state, allowing this to generate a policy tree.And, as far as we know, RTDP-bel and POMCP are the only POMDP [11] algorithms that allows to approximate a value function in any format.

This paper is organized as follows: firstly the MOMDP model for this application case is presented. After, the belief state GMM representation learning is discussed. Then, the two algorithms: (L)RTDP-bel and POMCP-based are proposed; the results are shown in order to compare their performances. Finally, future work is discussed.

II. UAV Safe Path Planning Problem

This paper addresses a problem of finding a navigation and guidance strategy (path and modes) which makes

*This work was not supported by any organization

[1]Jean-Alexis Delamer, Information Processing and Systems Departement (DTIS), ONERA, Toulouse, France jean-alexis.delamer@onera.fr

[2]Yoko Watanabe, Information Processing and Systems Departement (DTIS), ONERA, Toulouse, France yoko.watanabe@onera.fr

[3]Caroline P. Carvalho Chanel, Design and Control of Aerospace Vehicles Departement (DCAS), ISAE-SUPAERO - University of Toulouse, France caroline.chanel@isae.fr

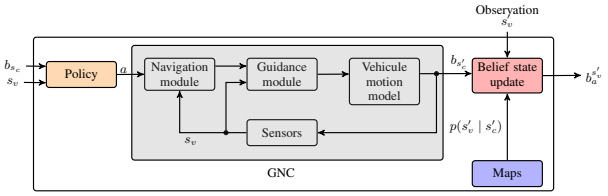


Fig. 1: System architecture diagram. The GNC closed-loop vehicle model is incorporated into the MOMDP transition function. Apriori information forms a set of probability grid maps of the environment and the sensor availabilities.

autonomous vehicles reach a given destination safely and efficiently in a cluttered environment. Autonomous vehicles are equipped with different sensors, such as INS and GPS, which are used in its GNC system to execute a path (Fig. 1). The work here presented can be applied to any type of autonomous vehicles, as long as their GNC model is well-defined. To illustrate the approach, this paper focuses on the UAV model proposed by [10], where the GNC closed-loop system is modeled as a transition function of the continuous vehicle state vector x (position, velocity, etc.). The navigation filter estimates this state x and its error covariance P for a selected navigation mode (or sensor). The guidance and control module executes a selected path by using (or not, depending on a selected guidance mode) the navigation solution. The execution precision is given by the execution error covariance Σ, which may depend on P. 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. These maps are used in planning task to predict the path execution accuracy, and then to evaluate obstacle collision risk with respect to it given a path.

III. MOMDP Model

The Mixed Observability Markov Decision Process (MOMDP) proposed by [3] and [16] is a variant of the POMDP (Partially Observable Markov Decision Process). The state is not partially observable, but a part of the state is known at each epoch. In this problem, an UAV always knows the current sensor availabilities which are considered as a part of the state. Consequently, MOMDP is applied to model this problem. But, in contrast to a classical MOMDP model [16], there is no partial observable state in this application case. The vehicle state vector x is unobservable from the planning model point of view, since there is neither partial nor direct observation on it. The only outputs considered from the GNC closed-loop model is the localization and execution error covariances P and Σ. Figure 1 illustrates the system architecture with different modules.

The MOMDP is defined as a tuple $\{\mathcal{S}_v, \mathcal{S}_c, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{C}, b_0\}$, where $\mathcal{S}_v$ is the bounded set of fully observable states; $\mathcal{S}_c$ is the bounded set of non observable continuous states; $\mathcal{A}$ is the bounded set of actions; Ω is the bounded set of observations; $\mathcal{T}$ is the state transition function; $\mathcal{O}$ is the observation function such as : $\mathcal{O}(o, a, s'_c, s'_v) = p(o|s'_c, s'_v, a) = 1$ if $o = s'_v$, or 0 otherwise; $\mathcal{C} : \mathcal{B} \times \mathcal{B} \times A \rightarrow \mathbb{R}$ is the cost function, with $\mathcal{B}$, the belief state space defined over $|\mathcal{S}| = |\mathcal{S}_v| \times |\mathcal{S}_c|$; and $b_0 = (s^0_v, b^0_{\mathcal{S}_c})$, where $b^0_{\mathcal{S}_c} \in \mathcal{B}_c$ is the intial probability distribution over the non observable continuous states, conditioned to $s^0_v \in S_v$, the initial fully observable state.

The visible state $s_v \in \mathcal{S}_v$ is defined as a tuple containing the fully observable booleans of the sensor availability $[0; 1]$, with N the number of sensors, the boolean on the collision, and the P the localization error covariance matrix propagated by the navigation module in function of a selected navigation sensor/mode in a given decision step. The s_v is define such as $s_v = \{F_{\text{sensor1}}, \ldots, F_{\text{sensorN}}, F_{\text{Col}}, P\}$. It is assumed that the collision flag F_{Col} is observable either by measuring or estimating a force of contact.

The non observable continuous state $s_c \in S_c$ is defined such as $s_c = x$, recalling that x is the continuous vehicle state vector (position, velocity, etc.).

An action $a \in A$ is defined as a tuple $\{d, m_n, m_g\}$: the discretized path direction $d \in D$; the navigation mode $m_n \in \mathcal{M}_n$ and the guidance mode $m_g \in \mathcal{M}_g$.

The transition function $T(s_v, s'_v, a, s'_c, s_c)$ is composed of two functions: a transition function $T_{\mathcal{S}_c}$ such as:

$$T_{\mathcal{S}_c}(s_c, s_v, a, s'_c) = f_{s'_c}(s'_c|s_c, s_v, a) \sim N(\bar{s'_c}, \Sigma'(s_v)),$$

which is based on the GNC closed-loop model, given that the probability distribution of a predicted state s'_c follows a normal distribution $N(\bar{s'_c}, \Sigma'(s_v))$, which in turn, is a function of the previous state s_c and the action a; and a transition function $T_{\mathcal{S}_v}$ such as $T_{\mathcal{S}_v}(s'_c, s'_v) = p(s'_v|s'_c)$, which represents the transition to s'_v and depends on the sensor availability maps and therefore depends only on the next state s'_c. Concretely,

$$T_{\mathcal{S}_v}(s'_v|s'_c) = \prod_{i=1}^{N+1} p(s'_v(i)|s'_c) \tag{1}$$

where N is the number of sensors, thus $N+1$ is the number of flags (booleans) in s_v, and $s'_v(i)$ the i-th flag. Then, the transition function becomes:

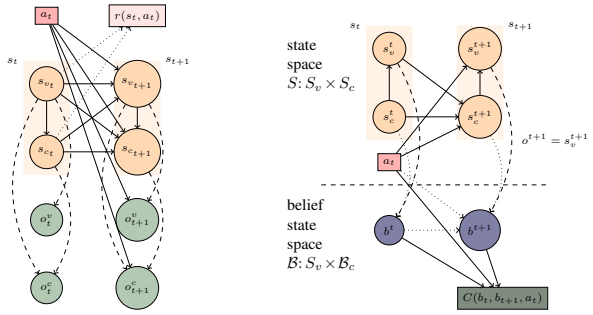
$$\begin{aligned} T(s_v, s'_v, a, s'_c, s_c) &= T_{\mathcal{S}_c}(s_c, s_v, a, s'_c) \times T_{\mathcal{S}_v}(s'_c, s'_v) \\ &= p(s'_v|s'_c) f_{s'_c}(s'_c|s_c, s_v, a) \end{aligned} \tag{2}$$

Note the execution error covariance matrix Σ from the GNC transition model represents the uncertainty envelope of the unobservable state s_c (represented by b_{s_c} in the Fig. 1), instead of P, the localization error covariance matrix. The belief state is updated after an action a followed by a perceived visible state $o' = s'_v$. The belief state update is decomposed into two functions. The first one corresponds to the GNC closed-loop transition function belief propagation:

$$b_{s'_c}(s'_c) = \int_{\mathcal{S}_c} f_{s'_c}(s'_c|s_c, s_v, a) b_{s_c}(s_c) ds_c \tag{3}$$

The second one is the probability of s'_v (given by the probability grid maps) that is computed based on $b_{s'_c}$:

$$p(s'_v|b, a) = \sum_{i=1}^{|G|} p(s'_v|s'_c \in c_i) p(s'_c \in c_i|b, a) \tag{4}$$



(a) Classical transition model for a MOMDP [3]

(b) proposed MOMDP transition model with $\Omega = S_v$.

Fig. 2: Difference between the transition model

where c_i corresponding to the i^{th} cell of the probability map and $|G|$ is the number of cells in the map. Finally, the belief state update function is defined as :

$$b'^{s'_v}_{s'_c,a}(s'_c) = \frac{p(s'_v|s'_c)\int_{\mathcal{S}_c} f_{s'_c}(s'_c|s_c, s_v, a) b_{s_c}(s_c) ds_c}{\sum_{i=1}^{|G|} p(s'_v|s'_c \in c_i) p(s'_c \in c_i|b, a)} \quad (5)$$

a) Why a MOMDP model instead of a classical POMDP model?: The choice of modelling this application case as a MOMDP instead a POMDP (Partially Observable Markov Decision Process) is twofold: (i) the MOMDP model offers the possibility of factorizing the state space, resulting the policy computation complexity, as the belief state probability distribution can be defined over a small belief state space $\mathcal{B}_c$ (which refers to the $\mathcal{S}_c$ space instead of the complete $\mathcal{S}$ space, such that $b = (s_v, b_{\mathcal{S}_c})$) – see Fig. 2 ; (ii) as the state space is factorized, one can factorize the observation space too. In this particular application case, the observation set which corresponds to S_c is empty. Therefore, the only observation set relies on S_v. Given that $\mathcal{O}(o, a, s'_c, s'_v) = p(o|s'_c, s'_v, a) = 1$ iff $o = s'_v$, or 0 otherwise, one can compute expectations directly over this variable (cf. Eq. 4 and 5).

A. Belief state representation learning

Figure 3a illustrates a belief state update step. The future belief state b_a is calculated after an action a from an initial belief state b (see also Fig 1). If b is Gaussian, b_a returned by the GNC transition function is also Gaussian. Then, in this example, an observation s_v is perceived, such that a probability $p(s_v|b_a) > 0$ only in the blue shaded cells. By using this observation and (Eq. 5), the belief state b_a is updated to $b_a^{s_v}$. The shape of this new belief is no longer Gaussian. Consequently, future belief states will not be Gaussian neither.

Gaussian belief states allow algorithmic simplifications. The Gaussian property allows planning algorithms to handle belief states directly, unlike previous works which approach belief states with Monte-Carlo particle filters [4], [17], thus reduces computation time. Therefore, this paper proposes to apply a machine learning technique to calculate a Gaussian Mixture Model (GMM) to approximate the new belief $b_a^{s_v}$.

The EM algorithm [2] is used in this work. EM learns a GMM belief state representation for a fixed number N_g of Gaussian functions. The ideal N_g is unknown; then it is necessary to compare the GMMs learned for different N_g's. So firstly, a sufficient number of samples s_c is generated from $b_a^{s_v}$. Then, for each N_g, the GMM is *trained* (it runs the EM algorithm). To decide on which GMM best fits the belief state, the Bayes Information Criterion (BIC) [2] is used, due to its interesting property of penalizing the complexity of the learned model (i.e., N_g). The GMM with smallest BIC score is selected. Note that the maximum N_g and the number of samples are problem-dependant and need to be tested empirically beforehand. Figures 3b and 3c show an example of the GMM learning result, where the initial belief state b_a (black dotted ellipse) is updated by an observation s_v of non-collision (the white part of the figure 3b).Then EM is applied to learn GMMs with different N_g from 1 to 7, and that with $N_g = 2$ (shown in red and blue ellipses) was selected according to the BIC score comparison (Fig 3c).

B. Cost function

The GMM approximation allows us to analytically compute the cost $\mathcal{C}$ of the belief state transition. The cost function calculates the volume of the uncertainty corridor (see [18] for more details on this volume computation) between two belief states (ensuring the safety and efficiency of the path). A fixed collision penalty cost multiplied by collision probability is also added to the cost function. Then:

$$\mathcal{C}(b_t, b_{t+1}) = \sum_{g \in b_t} \sum_{g\prime \in b_{t+1}} U(g, g\prime) p(g\prime \mid b'^{s_v}_a, g) w(g) w(g\prime) + K \times s'_v(Collision)). \quad (6)$$

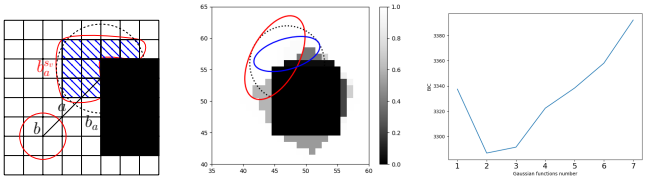
where b_t is the initial GMM belief, b_{t+1} is the learnt GMM belief, $U(g, g\prime)$ is the volume of the uncertainty corridor between two Gaussian functions from the mixtures, $p(g\prime \mid b'^{s_v}_a, g)$ the probability of being in $g\prime$ after an action a from g, $w(g)$ the weight of the Gaussian function provided by the EM algorithm and K the collision cost. Note the GMM approximation helps to compute the belief state transition cost (Eq. 6). The cost can be analytically computed, avoiding the application of particles filtering or Monte-Carlo cost evaluation.

C. Value function

The value function $V^\pi(b)$ is defined as the expected total cost (weighed by time using γ) the agent will receive from b_0 when following the policy π [11].

$$V^\pi(b) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[\mathcal{C}(b_t, b_{t+1}, \pi(b_t))\right] | b_0 = b \right]. \quad (7)$$

As the value function is built based on an expected sum of costs, one needs to pay attention to the form of the cost



(a) Deformation of the b_a during update.

(b) GMM learning result with $N_g = 2$.

(c) BIC score for different N_g's

Fig. 3: Example of the GMM learning algorithm result.

function. Also note that the cost function does not directly depend on the action, but this last has an indirect impact: the uncertainty of a state depends on the execution error covariance Σ affected by the navigation and guidance modes chosen. The optimal policy π^* is defined by the optimal value function $V^{\pi *}$, such as :

$$V^{\pi^*}(b) = \min_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[\mathcal{C}(b_t, b_{t+1}, \pi(b_t))\right] | b_0 = b\right] \tag{8}$$

Opening the sum in (Eq. 8), it holds to a Bellman's equation, which allows the application of dynamic programming. For example:

$$\begin{aligned} V(b) &= \min_{a \in A} \mathbb{E}\left[\mathcal{C}(b, b_a^{s_v}) + \gamma V(b_a^{s_v})\right] \\ &= \min_{a \in A} \sum_{s_v \in S_v} p(s_v|b,a)(\mathcal{C}(b, b_a^{s_v})) + \gamma V(b_a^{s_v})), \end{aligned} \tag{9}$$

when the value (Eq. 9) converges for all reachable belief states, within an ϵ error, one can extract the related optimized (partial-)policy [14].

IV. Algorithms

The cost function defined in (Eq. 6) depends on belief state transitions and is no more piecewise linear and convex (PWLC). In this case, the use of classical MOMDP algorithms, such as SARSOP [14] which uses α-vectors to represent the value function, is no more possible.

A. (L)RTDP-bel

The proposed algorithm is based on (L)RTDP and RTDP-bel, because RTDP-like algortihms do not require to have a PWLC value function. The idea is to directly evaluate the belief states (and not the states as in RTDP-bel) while exploring the convergence improvement of (L)RTDP.

Therefore, some functions and definitions need to be adapted. In particular, the residual function which calculates the difference between the value of the belief state and the result of the Q-value of b for a greedy action a. Then the residual is defined as :

$$R(b) = \left| V(b) - \min_{a \in A} \sum_{s_v \in S_v} p(s_v|b,a)(\mathcal{C}(b, b_a^{s_v})) + \gamma V(b_a^{s_v}) \right| \tag{10}$$

As in (L)RTDP, it is considered that a belief state has converged (and consequently solved being marked as *Labelled*) if the following definition is verified.

A value function $V(b)$ converges for a given belief state b relative to parameter $\epsilon > 0$ when $R(b) \leq \epsilon$.

(L)RTDP-bel, shown in Alg. 1, takes in entry an initial belief state b_0 and an ϵ parameter. While the initial belief is not solved, it will continue to simulate the greedy policy (performing trials). A trial in (L)RTDP-bel is very similar to the one of (L)RTDP, but there is an important difference (besides working with belief states instead of the states): during the update of the belief state of (L)RTDP-bel, the EM algorithm is used to learn a Gaussian mixture model to represent the belief state. When the goal is reached[1] the algorithm checks if each of the value of the belief states has converged (i.e solved). This check is done by the *Check-Solved* algorithm which does not differ from the Check-Solved algorithm of (L)RTDP. To obtain more details on these algorithms, please refer to [5], [6].

[1]it considers that it has reached the goal when the position of the goal belongs to the ellipsoid (3σ) defined by the current belief state.

Algorithm 1: (L)RTDP-bel

```
1  Function (L)RTDP-BEL(b_0,ε)
2      while b_0 not solved do
3          (L)RTDP-BEL-TRIAL(b_0,ε)
4      return π*_{b_0}
5  Function (L)RTDP-BEL-TRIAL(b_0,ε)
6      visited ⟵ ∅; b ⟵ b_0
7      while b not solved do
8          visited ⟵ b
9          if b ∉ Goal then
10             a_best ⟵ argmin_{a∈A} Q^{V_l}(b,a)
11             V_l(b) ⟵ Q^{V_l}(b, a_best)
12             b_a ⟵ execute a_best in b
13             s_v ⟵ sample s_v from p(s_v|b_a)
14             b_a^{s_v} ⟵ update(b_a, s_v)
15             b ⟵ b_a^{s_v}
16     while visited ≠ ∅ do
17         b ⟵ pop(visited)
18         if !CHECK-SOLVED(b,ε) then
19             break
```

B. POMCP

The POMCP algorithm [17] is a Monte-Carlo Tree Search algorithm for partially observable environments. POMCP works by sampling a state s in the current belief state (belief node or history h) and simulating sequences of action-observation (rollout procedure) to construct a tree, then calculates the average reward (or cost) for a belief node based on the average reward of children nodes. The algorithm keeps in memory the number of times a node was explored $N(h)$ and the number of times an action was chosen $N(ha)$ in this given node. As UCT [13], it applies the UCB1 greedy action selection strategy that is based on a combination of two characteristics: an approximation of the action's Q-value and a measure (given by $c\sqrt{\frac{\log N(h)}{N(ha)}}$) of how well-explored the action is, given this history (or belief node).

However, due to the particularities of the model addressed in this work and to the fact that this is a goal-oriented problem, the POMCP algorithm needs to be modified. Starting with an initial belief state b_0 (line 3), the algorithm (Alg. 2) will expands the tree for a given timeout duration. If the belief state is the goal (line 6), it returns, else it tests if the belief state is in the tree. If not, (line 8) the belief state is added. For each pair of action-observation, the next belief $b_a^{s_v}$ is also added to the tree (line 11). Note that, contrary to the classical POMCP algorithm, no state is sampled because the algorithm works directly on the belief state (specially for cost and Q-value functions computation). Thus, the next action is chosen using the UCB1 greedy action selection strategy (line 12). After, an observation s_v is sampled, and the belief state is updated (EM algorithm). The tree is expanded with this new belief state, and the Q-value (recursively) updated.

Algorithm 2: POMPC

Function *POMPC(b_0, c, γ)*
  while *!Timeout* **do**
    Expand(b_0, c, γ)
  $a^* \leftarrow \underset{a \in A}{\operatorname{argmin}}\, V(b_0)$

Function *Expand(b, c, γ)*
  if $b \in$ *Goal* **then**
    return 0
  if $b \notin T$ **then**
    for $a \in A$ **do**
      for $s_v \in S_v$ **do**
        $T(b_a^{s_v}) \leftarrow (N_{\text{init}}(b_a^{s_v}), V_{\text{init}}(b_a^{s_v}), \emptyset)$
  $\bar{a} \leftarrow \underset{a \in A}{\operatorname{argmin}}\, Q(b,a) - c\sqrt{\frac{\log N(b)}{N(b_{\bar{a}})}}$
  $s_v \sim \mathcal{G}(b_{\bar{a}})$ /* Random generator */
  $b_a^{s_v} \leftarrow update(b_{\bar{a}}, s_v)$
  $Expand(b_{\bar{a}}^{s_v}, c, \gamma)$
  $N(b) \leftarrow N(b) + 1$
  $N(b_{\bar{a}}) \leftarrow N(b_{\bar{a}}) + 1$
  $Q(b,\bar{a})' \leftarrow \sum_{s_v \in S_v} p(s_v|b,\bar{a})\,(C(b, b_{\bar{a}}^{s_v}) + \gamma V(b_{\bar{a}}^{s_v}))$
  $Q(b,\bar{a}) \leftarrow Q(b,\bar{a}) + \frac{Q(b,\bar{a})' - Q(b,\bar{a})}{N(b_{\bar{a}})}$
  $V(b) \leftarrow \min_{a \in A} Q(b,a)$

C. Belief state value initialization

As the MOMDP value function results from the application of dynamic programming minimization operator, the expert needs to ensure that the initial value of a belief state (or initialization heuristic value, as in Alg. 2 with V_{init}) must be an upper bound (or lower bound for a maximization operation) in order to preserve algorithm convergence - in particular the contraction property [8].

The belief state value initialization proposed in this paper explores the A* shortest path solution on the obstacle grid map. The execution error Σ is propagated over this path for the navigation and guidance modes with sensors most likely-available. Then the cost-minimizing navigation and guidance mode is selected among all the available modes. This value approximation gives a tighter upper bound than a worst-case navigation and guidance strategy.

V. Simulation results

The two MOMDP algorithms have been tested on a benchmarking framework for UAV obstacle field navigation [2] [15], which provides environments with different obstacle configurations. The UAV model (is the one model described in Section III). Here two different maps have been selected: "Cube baffle" which contains two cube obstacles and "Cube" which contain one cube obstacle both with a grid size of $100 \times 100 \times 20$. To perform these tests, it has been considered only two sensors onboard an UAV: INS and GPS. While INS is known to be available anywhere, a probability grid map of GPS availability was created based on a DOP map calculated by using a GPS simulator. For each test the initial belief state was $b_0 = (\bar{s}_c, \Sigma, s_v)$, where $\bar{s}_c = [5,5,7,0,0,0,0,0,0]$, $\Sigma = diag(1,1,4,0.01,0.01,0.04,0.01,0.01,0.01)$, $s_v = [1,1,0,P]$, with $P = \Sigma$ (note this is true only on initial belief state, even after the first action Σ and P differs) and the goal was in $(90,90,7)$. To obtain representative results, 1000 simulations have been run for each algorithm policy on two different maps. The parameters for the (L)RTDP-bel were $\gamma = 0.9, K = 1000$ and for the POMCP policy : $\gamma = 0.9, Time = 180min, c = 200, K = 1000$.

[2]benckmark framework from: www.aem.umn.edu/people/mettler/projects/AFDD/AFFDwebpage.htm

	CubeBaffle		Cube	
	(L)RTDP-bel	POMCP	(L)RTDP-bel	POMCP
Success Rate	96%	95%	96%	96%
Average cost	3393.84	3072.67	5892.91	4093.24
Average cost in %	0%	-9.47%	0%	-30.54%

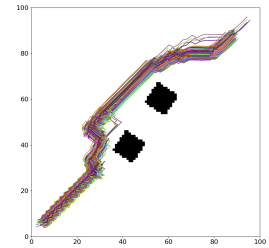
TABLE I: Performance comparison between (L)RTDP and POMCP.

The average simulation results are given in Table I. In terms of performance, the success rate is almost similar for each algorithm and map. However the average path cost is different between the algorithms, it can be seen that POMCP is more efficient than (L)RTDP. For the "Cube baffle" map the difference is smaller than for "Cube," POMCP reduces the cost of 9.45% for the first counter 30.54% to this last.

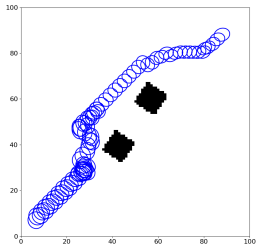
In Figure 4 some simulated paths are illustrated. The first column (Figures 4a, 4c, 4e and 4g) represent the simulated paths in 2D and the second column (figures 4b, 4d, 4f and 4h) represent the belief states of the most likely path. Regardless of the test map, the paths are perfectly within the bounds of the belief state calculated with the policies. The only exception is for the figures 4a and 4b, where different observations were perceived by the UAV during the simulations resulting in different paths. Thus for the sake of understanding only the belief states of the most likely path have been represented on the figure 4b. This supports the idea that representing belief states as GMM is a good idea to simplify cost computation and ensure the generalization of the MOMDP model with the GNC model.

Moreover, the paths simulated with the two algorithms are almost similar. This is because the test maps are simple and thus the shortest and safest path is straightforward to compute. However, it can be observed that the simulated trajectory with the POMCP policy is smoother. And more specifically with the "Cube" map, the POMCP has better-anticipated sensor availability and collision risk than (L)RTDP-bel. It was expected, because (L)RTDP-bel is greedier during policy computation and do more local optimization. (L)RTDP-bel optimizes better for belief states are closer to the obstacle than the POMCP which explores more uphill in the search tree. In the "Cube" map case the path simulated with the POMCP policy starts to avoid the obstacle much sooner resulting in lesser cost than the paths simulated with the (L)RTDP policy.

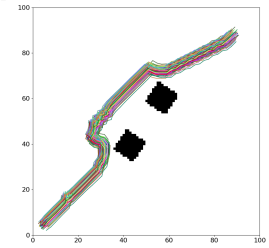
The 2D representation gives a good idea of the paths simulated, but it is interesting to analyze them in a 3D perspective. Figure 5 presents the simulated paths on the "Cube baffle" map Fig. 5a shows the paths obtained with the (L)RTDP-bel policy and Fig. 5b those with the POMCP policy. The first consideration is the paths are more scattered in height than in width. It is expected in these tests because GPS uncertainty is four time higher on height than on the other axes. It explains why the policies computed do not push the UAV to go above the obstacles because the maps are limited in height. Another consideration is that the POMCP



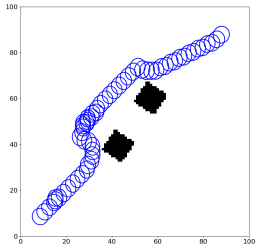
(a) Path simulated with the (L)RTDP policy on "Cube baffle".



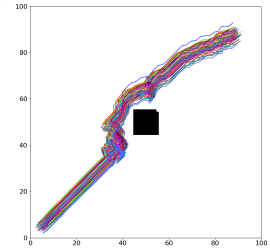
(b) Beliefs state representation of the (L)RTDP results.



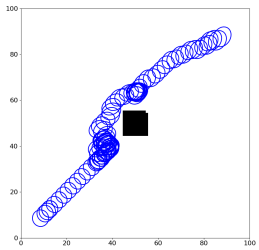
(c) Path simulated with the POMCP policy on "Cube baffle".



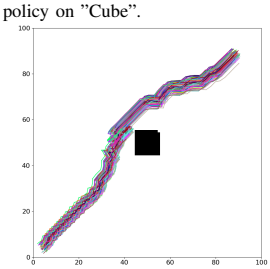
(d) Beliefs state representation of POMCP results.



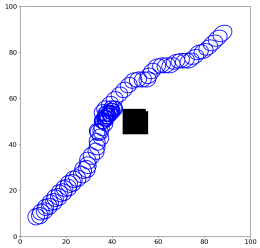
(e) Path simulated with the (L)RTDP policy on "Cube".



(f) Beliefs state representation of the (L)RTDP.



(g) Path simulated with the POMCP policy on "Cube".



(h) Beliefs state representation of POMCP results.

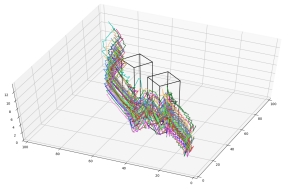
Fig. 4: Path simulated on the two maps for each algorithms.

policy (Fig. 5b) anticipate uncertainty around the obstacle by avoiding it and getting high.

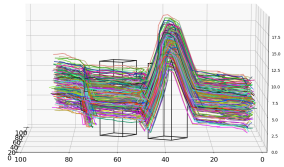
From these results, it is possible to say that both algorithms compute an acceptable policy. But POMCP has calculated a more effective policy due to its better convergence properties [17].

VI. Conclusion and Future Work

This paper presented a MOMDP model to solve safe path planning problem in urban environments, a belief state representation as Gaussian Mixture Models is presented, two algorithms are proposed, and results are compared. This problem can be viewed as a large MOMDP domain, as the non-observable state is continuous. Even when considering a discrete set of actions the problem is complex. The current results show that with goal-oriented algorithms it is possible to obtain significant results on simple maps. More evaluations are necessary, especially on real urban environments



(a) Path simulated with the (L)RTDP policy on "Cube baffle".



(b) Path simulated with the POMCP policy on "Cube baffle".

Fig. 5: 3D representing of the path simulated with each policy on the "Cube baffle" map.

[15].

Further work will include an extension to deal with a continuous action space. It is also planned to apply learning algorithms for value function and policy representation. So that, one can generalize the value (or policy) for belief states not visited during the optimization phase. Gaussian Process Dynamic Programming [9], for example, could be a potential solution to this interesting point.

References

[1] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart. Motion- and uncertainty-aware path planning for micro aerial vehicles. *Journal of Field Robotics*, 2014.

[2] D. W. Andrews and B. Lu. Consistent model and moment selection procedures for gmm estimation with application to dynamic panel data models. *Journal of Econometrics*, 2001.

[3] M. Araya-López, V. Thomas, O. Buffet, and F. Charpillet. A closer look at momdps. In *22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2010.

[4] H. Bai, D. Hsu, W. S. Lee, and V. A. Ngo. Monte carlo value iteration for continuous-state pomdps. In *Workshop on the algorithmic foundations of robotics*, 2010.

[5] B. Bonet and H. Geffner. Labeled rtdp: Improving the convergence of real-time dynamic programming. In *ICAPS*, 2003.

[6] B. Bonet and H. Geffner. Solving pomdps: Rtdp-bel vs. point-based algorithms. In *IJCAI*, 2009.

[7] A. Bry and N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[8] O. Buffet and O. Sigaud. Processus décisionnels de markov en intelligence artificielle, 2008.

[9] M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 2009.

[10] J.-A. Delamer, Y. Watanabe, and C. P. C. Chanel. Towards a momdp model for uav safe path planning in urban environment. In *IMAV 2017*, 2017.

[11] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 1998.

[12] 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*. Springer, 2009.

[13] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *ECML*, 2006.

[14] H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[15] B. Mettler, Z. Kong, C. Goerzen, and M. Whalley. Benchmarking of obstacle field navigation algorithms for autonomous helicopters. In *66th Forum of the American Helicopter Society:" Rising to New Heights in Vertical Lift Technology", AHS Forum 66*, 2010.

[16] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 2010.

[17] D. Silver and J. Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, 2010.

[18] Y. Watanabe, S. Dessus, and S. Fabiani. Safe path planning with localization uncertainty for urban operation of vtol uav. In *AHS Annual Forum*, 2014.