PAIS 2022
A. Passerini and T. Schiex (Eds.)

doi:10.3233/FAIA220065

Learning Path Constraints for UAV Autonomous Navigation Under Uncertain GNSS Availability

Marion ZANINOTTI [a,b], Charles LESIRE [b], Yoko WATANABE [b], Caroline P. C. CHANEL [a]
[a] *ISAE-SUPAERO, University of Toulouse, France*
[b] *ONERA, The French Aerospace Lab, France*

This paper addresses a safe path planning problem for UAV urban navigation, under uncertain GNSS availability. The problem can be modeled as a POMDP and solved with sampling-based algorithms. However, such a complex domain suffers from high computational cost and achieves poor results under real-time constraints. Recent research seeks to integrate offline learning in order to efficiently guide online planning. Inspired by the state-of-the-art CAMP (Context-specific Abstract Markov decision Process) formalization, this paper proposes an offline process which learns the path constraint to impose during online POMDP solving in order to reduce the policy search space. More precisely, the offline learnt constraint selector returns the best path constraint according to the GNSS availability probability in the environment. Conclusions of experiments, carried out for three environments, show that using the proposed approach allows to improve the quality of a solution reached by an online planner, within a fixed decision-making timeframe, particularly when GNSS availability probability is low.

online path planning, learning for planning, POMDP

1. Introduction

Solving autonomous navigation problems consists in finding a path from an initial position to a goal with a maximum efficiency, while avoiding the obstacles. These problems become challenging when the state of the vehicle is uncertain. Particularly, most of Unmanned Aerial Vehicles (UAVs) are equipped with a Global Navigation Satellite System (GNSS) receiver as navigation system. In an urban environment, the GNSS satellites can be masked by the buildings surrounding the UAV, the accuracy or even the availability of the measured position can then be significantly altered, what can lead to a fatal collision.

[1] formalize the UAV urban navigation problem under uncertain GNSS availability as a Partially Observable Markov Decision Process (POMDP) [2]. The latter is a principled approach to solve planning problems under uncertainty. However, POMDP planning faces two notorious problems. The first one is the *curse of dimensionality*: the size of the belief state space grows up exponentially with that of the state space. The second problem is the *curse of history*: the number of action/observation sequences

to evaluate during research grows up exponentially with the planning horizon [3]. The use of a Partially Observable Monte-Carlo Planning (POMCP) [4] algorithm makes it possible to overcome these difficulties. Nevertheless, the performance remains dependent on the search depth reached within the planning horizon, which is itself dependent on the *branching factor* of the search tree [5]. The branching factor includes both the *action factor*, *i.e.* the number of actions available in each belief state, and the *stochastic factor*, *i.e.* the number of possible observations for each action. The stake is then to reduce the branching factor in order to scale up planning. This is all the more important for online planning: the planner has to make a decision quickly whereas the long-planning horizon of such a real-world task incurs prohibitive computational cost. For that, incorporating domain abstraction is a promising approach. [6] introduce Context-specific Abstract Markov Decision Process (CAMP), an abstraction of the original MDP model, obtained by imposing the *best constraint* on the states and actions considered by the agent. This best constraint is chosen by an offline learnt *context selector* according to the *features* of a task.

Inspired by this CAMP domain abstraction, this paper proposes to learn offline the context selector and to impose the best constraint returned in order to reduce the policy search space during online POMDP solving, for the UAV urban navigation problem. The context selector chooses the constraint which reduces the UAV position state space while preserving the solution optimality, in function of the GNSS availability probability map of a task. Unlike the original CAMP, we address a partially observable domain. In our case, applying action space abstraction is not straightforward as states are not fully observable. Nevertheless, a state space abstraction can be achieved through modification of the cost function for penalizing the constraint violation, which will modify the action outcomes. Additionally, as our objective is to perform online planning for the UAV safe navigation problem, whereas an offline POMCP variant is used in the offline process, an online version is applied for planning. As result, we investigate the use of different algorithms for learning and planning, which has not been done in the original work of CAMP. Thus, regarding the UAV navigation problem with uncertain GNSS availability, our contribution is twofold: (i) we investigate if state abstraction, by adapting the CAMP framework for a partially observable domain, gives better results when compared to a full POMDP model, and (ii) we evaluate if such a CAMP-inspired approach is robust if we use a different algorithm for learning and planning.

After providing the theoretical background and the related work in the next section, we present the CAMP method adapted to our problem in Section 3. Experimental results are reported in Section 4, demonstrating the performance improvement of online planning. Finally, Section 5 includes concluding remarks and future works.

2. Background and Related Work

2.1. POMDP Preliminaries

A POMDP [2] is defined as a tuple $(\mathscr{S}, \mathscr{A}, \Omega, \mathscr{T}, \mathscr{O}, \mathscr{C}, b_0, \gamma)$, where $\mathscr{S}$, $\mathscr{A}$, and Ω denote respectively spaces of states, actions, and observations. The *transition function* $\mathscr{T}(s,a,s') = p(s'|s,a)$ represents the dynamics of the agent as the probability of transiting from s to s' by taking action a. The *observation function* $\mathscr{O}(a,s',o) = p(o|s',a)$

specifies the probability of observing o after taking action a to reach s'. The *cost function* $\mathscr{C}(s,a)$ defines the cost of taking action a in s. b_0 denotes the initial belief state, and $\gamma \in [0,1]$ is a discount factor expressing a preference for minimizing immediate over future cost.

POMDPs capture partial observability of the system using the *belief state* b, *i.e.* a probability distribution over $\mathscr{S}$, which is updated after each action a and observation o using the Bayes' rule. A POMDP policy $\pi : \mathscr{B} \to \mathscr{A}$ prescribes an action for each belief state in the belief space $\mathscr{B}$. Solving a POMDP requires to find the *optimal policy* π^* minimizing the expected future cost, called the *value*, for all $b \in \mathscr{B}$. The value of the policy π^* in belief b is defined as:

$$V^{\pi^*}(b) = \min_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathscr{C}(b_t, \pi(b_t)) \,\middle|\, b_0 = b\right] \tag{1}$$

Additionally, the Q-value of an action a in belief b can be defined as:

$$Q^{\pi}(b,a) = \mathbb{E}\left[\mathscr{C}(b,a) + \sum_{t=1}^{\infty} \gamma^t \mathscr{C}(b_t, \pi(b_t))\right] \tag{2}$$

2.2. UAV Urban Navigation POMDP-based Problem

The original planning model, proposed by [1], is formalized as a Mixed-Observability Markov Decision Process (MOMDP) [7], a special class of the POMDP framework. The state space is factorized into fully and partially observable state variables, respectively denoted by s_v and s_h, what reduces the belief state space dimension, and in turn, reduces policy computation time. The state tuple $s = (s_h, s_v) \in \mathscr{S}$ is defined with $s_h = (\mathscr{X}, \mathscr{V}, \beta_a)$ where $\mathscr{X}$ and $\mathscr{V}$ are the vehicle position and velocity, and β_a is the IMU acceleration measurement bias, and $s_v = (F_{col}, F_{GNSS}, P, t_{flight})$ with F_{col} and F_{GNSS} the collision and GNSS availability Boolean flags, P the error covariance matrix over s_h, and t_{flight} the flight time elapsed. An action $a \in \mathscr{A}$ corresponds to the desired velocity direction. The action space $\mathscr{A}$ is a finite set of 10 actions, following 8 radial directions in the 2D horizontal plane, plus up and down. An observation $o \in \Omega$ is defined as the sub-tuple $o = s_v$ of the state tuple, given a full observability of (F_{col}, F_{GNSS}) and a deterministic transition of (P, t_{flight}). This partial state observability limits the branching factor of the search tree. The transition function follows a GNC (Guidance, Navigation and Control) model, composed of the vehicle motion model, a guidance law, a state estimator, and the IMU and GNSS sensor models. The GNSS availability F_{GNSS} affects the error covariance matrix P, which affects the belief state b' after transition. In brief, P grows when GNSS is unavailable, resulting in more collision risk. Finally, the cost function is defined as:

$$\mathscr{C}(s,a) = \begin{cases} 0 & \text{if goal reached} \\ K_{col} - t_{flight} & \text{if collision} \\ \Delta T_a & \text{otherwise} \end{cases} \tag{3}$$

with $\Delta T_a > 0$: the action execution time, and $K_{col} > 0$: the cost penalty in case of collision. When a collision occurs, the cost is this penalty subtracted with the flight time elapsed. Added to the sum of the previous action execution times, all the collision paths are then equally penalized.

2.3. MinPOMCP-GO Algorithm

[1] propose the POMCP - Goal-Oriented (POMCP-GO) algorithm, an offline goal-oriented variant of POMCP [4]. It samples a state s from the initial belief state b_0, corresponding to the *root node*, and simulates action/observation sequences, through *trials*, in order to evaluate actions while building a tree of nodes. To perform a trial, POMCP-GO follows a given action selection strategy and a heuristic node value initialization. For the action selection, it relies on the Upper Confidence Bounds (UCB1) strategy [8] to deal with the exploration–exploitation dilemma. A trial is stopped when a terminal state is reached (a goal or collision state), and this procedure is repeated during a fixed timeframe.

Each tree node h represents a history of action/observation sequences from the initial belief state. The Q-value (Eq. 2) of an action a in a belief state is approximated by $Q(h,a)$, which is the mean cost returned from all trials started from h when a was selected. This approximation incurs a well-known bias, which decreases as the number of trials increases [9]. To accelerate the policy value convergence by reducing the Q-value bias, [10] propose the MinPOMCP-GO algorithm which uses a *Min-Monte-Carlo backup* [9].

The present paper approach is based on this MinPOMCP-GO algorithm. During tree building, MinPOMCP-GO initializes the Q-value of a newly created node to a pre-computed heuristic value, corresponding to the flight time left to the goal estimated by the Dijkstra algorithm [11]. Even if this heuristic function is more informative than the traditional rollout used in POMCP, it does not take into account GNSS availability probability. The latter is only indirectly considered by back-propagating the cost penalty of a collision due to the uncertain estimated UAV position.

2.4. Domain Abstraction

Sampling-based algorithms, such as POMCP and variants, suffer from exponential complexity with respect to the branching factor of the search tree. In our UAV navigation problem under uncertain GNSS availability, the solver cannot explore enough, within a short decision-making timeframe, to prevent collisions. In difficult environments, with obstacles reducing GNSS availability probability, navigation mission safety may be compromised. So, we focus on incorporating domain abstraction to reduce the branching factor and thus to improve online planning solutions.

State Aggregation. One well-known technique of domain abstraction is *state aggregation*: the state space is reduced by clustering equivalent states, *i.e.* states that share some fully-identical properties - *exact aggregation* - or nearly-identical properties - *approximate aggregation* - and treating each of these resulting state clusters as one. In [12], the authors list the existing methods of exact state aggregation and unify them to deduce five generic functions. However, since two states rarely share some fully-identical properties, exact abstraction is often useless, while approximate abstraction can achieve greater degrees of compression. In [13], the authors present four types of approximate aggregation and demonstrate that they lead to a bounded loss of optimality of behavior. In [5], the authors generalize the formulation of two of these four types of aggregation and apply them to Monte-Carlo Tree Search (MCTS). AS-UCT [14], ASAP-UCT [15], and OGA-UCT [16] are other implementations of state or state-action pair aggregation

to UCT, a MCTS algorithm variant. All of these methods have not been applied in the partially observable framework.

Hierarchical Planning. Another approach to domain abstraction is *hierarchical planning*. It consists in decomposing the planning problem into a network of independent subgoals. *Hierarchical Dynamic Programming* (HDP) [17] is an example of hierarchical planning for navigation problems. A hierarchy of MDPs is constructed and solved using a hierarchical variation of value iteration. *Abstract Markov Decision Process* (AMDP) [18] is a more general method, which allows any MDP planner to be used. Both HDP and AMDP are *top-down* approaches: they select the subgoal before performing planning to reach it. Contrary to *bottom-up* approaches, they present the advantage that planning is necessary only for subgoals used for task completion. Nevertheless, the way to define appropriate subgoals remains an open question.

Integrating Learning for Planning. A third method is to integrate an offline learning phase as a first step, to guide the search during online planning. The CAMP approach [6], which has inspired this paper, is part of this category. It reduces the state and action spaces of a MDP by imposing a constraint learnt according to the features of a task. Another example is Macro-Action Generator-Critic (MAGIC) [19], a kind of *temporal abstraction*, which learns the more efficient set of candidate macro-actions to cut down the effective planning horizon for online POMDP planning.

As previously discussed in Section 2.3, GNSS availability probability is only taken into account by back-propagating the cost penalty when a collision occurs. The planning efficiency can hence be improved by using this information to further focus the search on more relevant areas, *i.e.* where GNSS is more likely available. For this purpose, the CAMP method seems a good candidate to leverage. Implementing a similar approach for our problem allows to reduce the UAV position state space in function of a probability map of GNSS availability, considering the latter as a task feature.

3. Learning Path Constraints based on GNSS Availability

3.1. Approach Overview

The objective of the CAMP method [6] is to learn a context selector $f : \Theta \rightarrow \mathscr{C}$. Each *training task* corresponds to a *feature vector* $\theta \in \Theta$. For each feature vector, the best constraint $C^* \in \mathscr{C}$ is identified. The pairs (θ_i, C_i^*) are given to a neural network to learn f. Once the context selector f is learnt, the best constraint C^* returned from the feature vector θ of a *test task* is then imposed to guide online planning.

In our navigation problem under uncertain GNSS availability, we assume a given environment, *i.e.* known obstacles on a map, and a given navigation mission, *i.e.* fixed initial position and goal. Figure 1 describes our application of the context selector learning process to our problem. The probability maps of GNSS availability are used as feature vectors. For each training map of GNSS availability probability, the best constraint is identified. We define a constraint as a corridor of the environment in which the UAV must stay, which we evaluate by performing planning within a *training timeout*. Then, these probability maps of GNSS availability and the associated best constraints are

used to learn the context selector. Finally, the test tasks are solved online, imposing the best constraints returned by the context selector from the test maps of GNSS availability probability. Each step of this process is detailed in the following sub-sections.

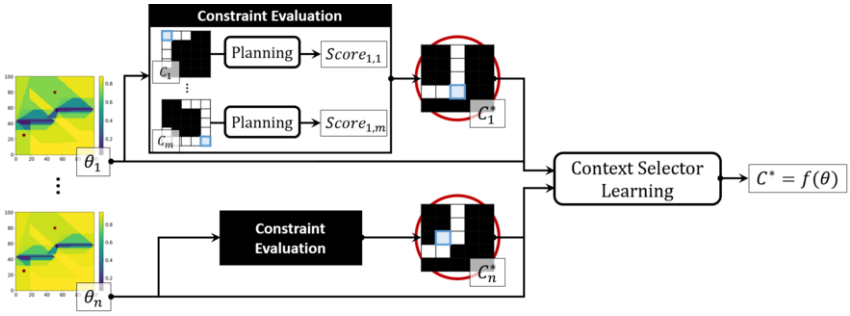

Figure 1. Learning the context selector f by identifying the best constraint C_i^*, for each training task i associated to a probability map of GNSS availability θ_i.

3.2. Feature Vectors

As previously discussed, GNSS availability is crucial to determine safe paths for our UAV. As the GNSS satellites are orbiting around the Earth, the GNSS availability probability varies with the time-of-the-day even for a fixed obstacle environment. We then propose to compute path constraints based on probability maps of GNSS availability.

The expected GNSS position accuracy is given by the *Position Dilution Of Precision* (PDOP) metric. Given the satellite geometry, user location and surrounding environment, a PDOP map is generated by using a GNSS simulator. We assume PDOP value to follow a zero-mean Gaussian distribution [1]. Then, the PDOP map is transformed into a probability map of GNSS availability by setting a maximum position error threshold ε:

$$\Pr(F_{GNSS} = 1) = \mathbf{erf}\left(\frac{\varepsilon}{\sqrt{2}\mathrm{PDOP}}\right). \tag{4}$$

with **erf**: the Gauss error function.

We generated test task features by setting different ε values to cover the easy and difficult cases where GNSS is most-like available/unavailable. Then, the training task features were generated by linear combination of these test task features with randomly selected coefficients, for more feature variety.

3.3. Constraint Definition and Evaluation

Constraint Definition. We divide the environment map into $n_L \times n_l \times n_h$ areas in an uniform way. n_L denotes the number of areas over the length, n_l over the width, and n_h over the height. For each of these areas, we define a corridor of areas leading from the initial position to the goal, passing through this area, called *passage area*. For that, we concatenate the paths resulting from the A* algorithm [20] from the area including the initial position to this passage area, and from the latter to the area including the goal

position. We use the number of areas constituting the path as cost function in the A* algorithm. We obtain thus at most $n_L \times n_l \times n_h$ different corridors of areas, corresponding to *candidate constraints*, in which the UAV is allowed to navigate. The sub-figure in the middle of Figure 2 shows a candidate constraint defined by dividing the environment map into $(n_L = 5) \times (n_l = 5) \times (n_h = 1)$ areas, and using the top left area as passage area, which is highlighted in blue.

Planning with Constraint. For each training map of GNSS availability probability, all the candidate constraints are evaluated. For that, offline planning imposing the candidate constraint is performed. We use the MinPOMCP-GO planning algorithm [10], adapting the heuristic function, which estimates the flight time left to the goal, so that the path constraint is respected. Figure 2 illustrates an example of the heuristic map obtained from a given environment, navigation mission, and candidate constraint. On the environment map on the left, as on the following maps, the initial position and the goal are respectively represented by a point and a star, and the obstacles are depicted in yellow. On the heuristic map on the right, the estimated flight time left to the goal is represented inside the constraint.

To impose a constraint, the cost function of the planning model (Eq. 3) is also adapted so that it considers a violation of the constraint as a terminal state which leads to a cost penalty K_{constr}. In addition, the collision cost is saturated by a minimal threshold $K_{col_{thr}}$, as some imposed constraints incur long flight times. The cost function then becomes:

$$\mathscr{C}(s,a) = \begin{cases} 0 & \text{if goal reached} \\ \max(K_{col} - t_{flight}, K_{col_{thr}}) & \text{if collision} \\ K_{constr} & \text{if constraint violation} \\ \Delta T_a & \text{otherwise} \end{cases} \tag{5}$$

Best Constraint Identification. In the original CAMP approach [6], a candidate constraint is evaluated using a score formulation, which expresses the trade-off between *how much planning is sped up* and *how much optimality is preserved* imposing this constraint. The planning time and the policy value are obtained as means over several online-computed paths. In our method, we perform offline planning to evaluate the candidate constraints. Hence, reaching the convergence on the policy value is required to estimate the planning time and the policy value. However, this convergence is difficult to judge and achieve it can take too long. Therefore, to evaluate a candidate constraint, we stop planning when a *training timeout* is reached, and we express the score as the opposite of the resulting initial belief state value $V^{\pi}(b_0)$. For a probability map of GNSS availability θ_i, the candidate constraint that achieves the highest score, *i.e.* the lowest initial belief state value, is chosen as the best constraint, and is noted C_i^*.

3.4. Context Selector Learning and Online Planning

The training maps of GNSS availability probability $\{\theta_i\}$ and the associated best constraints $\{C_i^*\}$ are used to train a neural network with cross-entropy loss, resulting in the context selector f (Fig. 1). The generic neural network available in the CAMP framework is applied, with the proposed Fully Connected Network architecture [6].

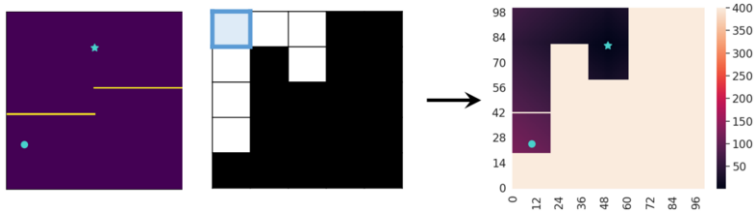

Figure 2. Generation of the heuristic map (right) from an environment, a navigation mission (left), and a candidate constraint (middle).

At test time, the best constraint is returned by the context selector, given the probability map of GNSS availability of the test task: $C^* = f(\theta)$. This constraint is then integrated in the model to reduce the UAV position state space during online planning, by imposing to compute navigation paths that stay within the constraint. We use two planning algorithms to compute these navigation paths. The first one is MinPOMCP-GO, also used for evaluating the candidate constraints in the training phase. The second algorithm is MinPOMCP-GO*: it is a variant of MinPOMCP-GO in which trials end whenever a previously unvisited leaf node is encountered instead of ending a trial only when a terminal state is reached. MinPOMCP-GO* is aimed to be used online, as it produces more trials with a shortest depth, hence favoring short-term performance that would help avoiding collisions.

4. Experiments

We implement the previously described method to three navigation benchmark environments available in [21]: *Cube Baffle*, containing two cubes, *Wall Baffle*, containing two walls, and the real downtown of *San Diego*. They are illustrated in Figure 3.

(a) (b) (c)

Figure 3. *Cube Baffle* (a), *Wall Baffle* (b), and *San Diego* (c).

To evaluate our approach, four test tasks are solved for each environment, numbered from 1 to 4, corresponding to the maps presenting from the lowest to the highest GNSS availability probabilities. We compare the results obtained imposing the best constraint returned by the context selector with those obtained without constraint. The performance metrics are the number of collisions and the mean cost obtained considering a fixed decision-making timeframe. The lower they are, the better performance is.

4.1. Material

To carry out the experiments, we use a supercomputer constituted of 24 cores. For each of these cores, the frequency is of 2.60 GHz, the Random Access Memory size is of 96 Gb, and the cache size is of 19.25 Mb.

4.2. Settings

In the following, we describe the settings used in our experiments. The GNC model and the reference velocity settings are the same as those described in [1].

Initial Position and Goal. The mean initial position is set to X_0 = [10, 25, 5] for *Cube Baffle* and *Wall Baffle*, and X_0 = [110, 60, 5] for *San Diego*. The goal position is set to X_G = [85, 78, 5] for *Cube Baffle*, X_G = [50, 80, 5] for *Wall Baffle*, and X_G = [200, 125, 5] for *San Diego*.

Map Decomposition. The map size of *Cube Baffle* and *Wall Baffle* is [101, 101, 21]. For *San Diego*, it is [217, 167, 24]. The maps are uniformly divided into $(n_L = 5) \times (n_l = 5) \times (n_h = 1)$ areas.

Model and Solver. The factor γ is set to 1, and the action cost ΔT_a is set to 2.2. The collision penalty K_{col}, its threshold $K_{col_{thr}}$, and the constraint violation penalty K_{constr} (Eq. 5) are respectively set to 450, 350, and 450. The exploration coefficient c of UCB1 is set to 6.

Training Tasks. The training timeout is set to 2 minutes and the number of training tasks, *i.e.* the number of probability maps of GNSS availability used for training, is 30.

Neural Network and Test Tasks. The neural network loss threshold is set to 1.8. The decision-making timeframe is set to 2 seconds and the number of test tasks, *i.e.* the number of probability maps of GNSS availability used for testing, is 4. These maps are generated with the error thresholds: ε = 1, 2, 5, and 10 meters. For each test task, 50 episodes are launched.

4.3. Results

The performance metric values obtained for each environment are summarized in Table 1. The probability maps of GNSS availability at the initial and goal altitude are displayed as background of the following figures, the resulting paths are plotted in red and the collisions are represented by black dots.

		MinPOMCP-GO						***MinPOMCP-GO****					
		No constraint		**Constraint**		**Relative Gain (%)**		**No constraint**		**Constraint**		**Relative Gain (%)**	
		N_{col}	*Cost*	N_{col}	*Cost*	N_{col}	*Cost*	N_{col}	*Cost*	N_{col}	*Cost*	N_{col}	*Cost*
Cube Baffle	1	2	115.144	0	114.296	100.00	0.74	0	91.912	0	106.072	/	-15.41
	2	0	96.936	0	97.808	/	-0.90	0	93.592	0	95.144	/	-1.66
	3	1	104.360	0	105.688	100.00	-1.27	2	108.848	0	100.720	100.00	7.47
	4	1	102.648	0	98.384	100.00	4.19	0	94.824	0	99.368	/	-4.79
Wall Baffle	1	21	243.848	6	162.936	**71.43**	**33.18**	12	179.696	0	116.664	**100.00**	**35.08**
	2	14	191.600	3	141.616	**78.57**	**26.09**	9	152.024	0	116.720	**100.00**	**23.22**
	3	0	85.336	0	95.632	/	-12.07	0	84.816	0	95.768	/	-12.91
	4	3	110.976	2	102.616	33.33	7.53	1	95.216	0	87.280	100.00	8.33
San Diego	1	37	387.776	36	370.904	2.70	4.35	40	385.800	35	355.352	12.50	7.89
	2	39	381.368	31	347.800	**20.51**	8.80	27	305.408	23	278.600	14.81	8.78
	3	34	355.872	18	249.392	**47.06**	**29.92**	32	334.264	8	180.576	**75.00**	**45.98**
	4	27	305.496	9	189.824	**66.67**	**37.86**	28	311.352	11	199.136	**60.71**	**36.04**

Table 1. Comparison of the performance metrics obtained by imposing the best constraint with the ones without constraint. Relative gain is computed as relative change, taking the performance metric value obtained without constraint as reference. The considerably performance gains are presented in bold.

For the *Cube Baffle* environment, the costs obtained without constraint and imposing the best constraint, using MinPOMCP-GO or MinPOMCP-GO*, are similar for all the test tasks. Indeed, the UAV does not fly close enough to the cubes and the GNSS availability probability is sufficiently high. Hence, very few collisions occur, even without constraint. Figure 4 shows the resulting paths without constraint and imposing the best constraint for the first test task, corresponding to the lowest GNSS availability probabilities. The imposed constraint makes the resulting paths deviate to avoid the zones of possible GNSS loss, to reduce the collision risk.

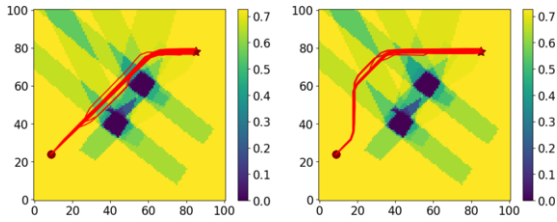

Figure 4. Results obtained for *Cube Baffle*, for test task 1: paths obtained using MinPOMCP-GO*, without constraint (a), and with the best constraint (b).

For the *Wall Baffle* environment, with MinPOMCP-GO or MinPOMCP-GO*, the number of collisions and the cost obtained imposing the best constraint are considerably lower than those without constraint for the test tasks (1) and (2), corresponding to the two maps presenting the lowest GNSS availability probabilities. For test task (1), using MinPOMCP-GO, the number of collisions obtained imposing the best constraint is reduced of almost 72%, and using MinPOMCP-GO*, it is reduced to 0. Figure 5 shows the resulting paths without constraint and imposing the best constraint. For these first two test tasks, the best constraint forces to fly over the wall, where GNSS availability probability is greater, instead of flying between the two walls as obtained when no constraint is imposed. Even if the flight time becomes a bit longer, the cost is much reduced because less collisions occur. That is, the mission safety is largely improved. For the third test task, using MinPOMCP-GO or MinPOMCP-GO*, the cost is slightly increased when imposing the best constraint, still favoring the safer paths flying over the wall. Finally, for the fourth test task, presenting the highest GNSS availability probabilities, the best constraint only imposes to slightly move away from the first wall. It results in a slight decrease of the number of collisions and the cost, with MinPOMCP-GO or MinPOMCP-GO*.

The *San Diego* environment includes multiple buildings, which incurs a lot of regions where GNSS availability probability is particularly low. Without constraint, the mission leads to a collision in most episodes, for each test task. The best constraints returned correspond to pass to the left of the obstacles (Fig. 6). With MinPOMCP-GO or MinPOMCP-GO*, the number of collisions and the cost are decreased imposing the best constraint, particularly for the two maps presenting the highest GNSS availability probabilities, test tasks (3) and (4). For the third test task, the cost is decreased to almost 46% using MinPOMCP-GO*, and for the fourth task, it is reduced to almost 38% using MinPOMCP-GO.

In conclusion, for the three environments, imposing the best constraint always reduces the number of collisions, with any MinPOMCP-GO variant. This gain on the number of collisions and the one on the cost are much greater for difficult environments,

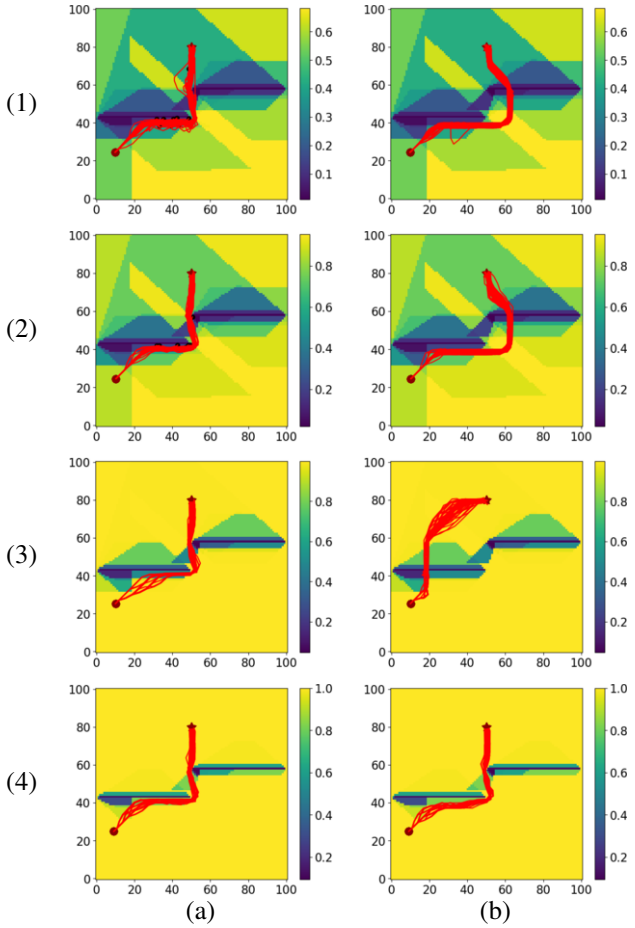

Figure 5. Results obtained for *Wall Baffle*: paths obtained using MinPOMCP-GO*, without constraint (a), and with the best constraint (b)

comprising multiple obstacles and presenting low GNSS availability probabilities. Moreover, although the context selector is learnt from MinPOMCP-GO, imposing the best constraint returned improves clearly the online planning performance even when using MinPOMCP-GO*.

5. Conclusion

In this paper, we have proposed a learning-based state abstraction approach to address a partially observable problem of UAV autonomous navigation, where the GNSS unavailability may have a dramatic impact on the UAV path. We have then implemented a process to learn the best path constraint, *i.e.* the best corridor in which the UAV must navigate, from a set of probability maps of GNSS availability. We have evaluated this approach on different environments, including a realistic urban one. The presented results have shown that first, imposing this best constraint can indeed improve the quality of the online-computed path, especially when uncertainty is high, and second, it achieves good

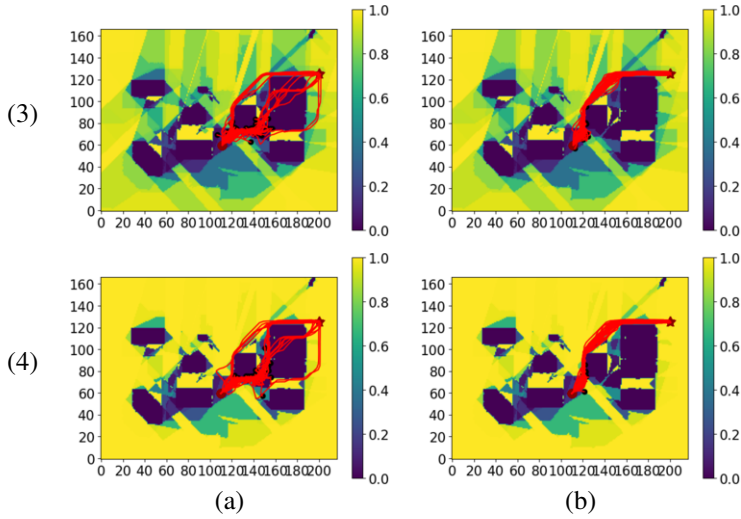

Figure 6. Results obtained for *San Diego*, for test tasks (3) and (4): paths obtained using MinPOMCP-GO*, without constraint (a), and with the best constraint (b)

performances although only the state space is abstracted, and different solvers are used for learning and planning.

Future works will generalize this approach by not only considering the probability map of GNSS availability as feature, but also the initial and goal positions. To do so, we will avoid to evaluate all the possible constraints by only considering the most suitable candidate constraints, in order not to generate a huge number of training data. For example, in our navigation problem, only three constraints may be considered for each feature vector: the one corresponding to the shortest path, the one maximizing GNSS availability probability, and the one weighting the both of them.

References

[1] Delamer JA, Watanabe Y, Ponzoni Carvalho Chanel C. Safe path planning for UAV urban operation under GNSS signal occlusion risk. Robotics and Autonomous Systems. 2021;142:103800.

[2] Kaelbling LP, Littman ML, Cassandra AR. Planning and acting in partially observable stochastic domains. Artificial Intelligence. 1998;101:99-134.

[3] Pineau J, Gordon G, Thrun S. Anytime Point-Based Approximations for Large POMDPs. Journal of Artificial Intelligence Research (JAIR). 2006;27:335–380.

[4] Silver D, Veness J. Monte-Carlo Planning in Large POMDPs. In: Advances in Neural Information Processing Systems (NeurIPS). Vancouver, BC, Canada; 2010. p. 2164–2172.

[5] Hostetler J, Fern A, Dietterich T. State Aggregation in Monte Carlo Tree Search. In: AAAI Conference on Artificial Intelligence (AAAI). Québec City, QC, Canada; 2014. p. 2446–2452.

[6] Chitnis R, Silver T, Kim B, Kaelbling L, Lozano-Perez T. CAMPs: Learning Context-Specific Abstractions for Efficient Planning in Factored MDPs. In: Conference on Robot Learning. London, UK; 2021. p. 64-79.

[7] Ong SCW, Png SW, Hsu D, Lee WS. Planning under Uncertainty for Robotic Tasks with Mixed Observability. The International Journal of Robotics Research. 2010;29(8):1053-68.

[8] Kocsis L, Szepesvári C. Bandit Based Monte-Carlo Planning. In: European Conference on Machine Learning (ECML). Berlin, Germany; 2006. p. 282-93.

[9] Keller T, Helmert M. Trial-Based Heuristic Tree Search for Finite Horizon MDPs. In: International Conference on Automated Planning and Scheduling (ICAPS). Rome, Italy; 2013. p. 135–143.

[10] Carmo AR, Delamer JA, Watanabe Y, Ventura R, Ponzoni Carvalho Chanel C. Entropy-based adaptive exploit-explore coefficient for Monte-Carlo path planning. In: International Conference on Prestigious Applications of Intelligent Systems (PAIS). (Digital ECAI); 2020. p. 1-8.

[11] Dijkstra EW. A Note on Two Problems in Connexion with Graphs. Numerische Mathematik. 1959;1:269–271.

[12] Li L, Walsh TJ, Littman ML. Towards a Unified Theory of State Abstraction for MDPs. In: International Symposium on Artificial Intelligence and Mathematics (ISAIM). Fort Lauderdale, FL, USA; 2006. p. 531-9.

[13] Abel D, Hershkowitz DE, Littman ML. Near Optimal Behavior via Approximate State Abstraction. In: International Conference on International Conference on Machine Learning (ICML). New York City, NY, USA; 2016. p. 2915–2923.

[14] Jiang N, Singh S, Lewis R. Improving UCT Planning via Approximate Homomorphisms. In: International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS). Paris, France; 2014. p. 1289–1296.

[15] Anand A, Grover A, Mausam M, Singla P. ASAP-UCT: Abstraction of State-Action Pairs in UCT. In: International Joint Conference on Artificial Intelligence (IJCAI). Buenos Aires, Argentina; 2015. p. 1509–1515.

[16] Anand A, Noothigattu R, Mausam, Singla P. OGA-UCT: On-the-Go Abstractions in UCT. In: International Conference on Automated Planning and Scheduling (ICAPS). London, UK; 2016. p. 29–37.

[17] Bakker B, Zivkovic Z, Kröse B. Hierarchical dynamic programming for robot path planning. In: International Conference on Intelligent Robots and Systems (IROS). Hamburg, Germany; 2005. p. 2756-61.

[18] Gopalan N, desJardins M, Littman ML, MacGlashan J, Squire S, Tellex S, et al. Planning with Abstract Markov Decision Processes. In: International Conference on Automated Planning and Scheduling (ICAPS). Pittsburgh, PA, USA; 2017. p. 480-8.

[19] Lee Y, Cai P, Hsu D. MAGIC: Learning Macro-Actions for Online POMDP Planning. In: Robotics: Science & Systems (RSS). (Held Virtually); 2021. .

[20] Hart PE, Nilsson NJ, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics. 1968;4(2):100-7.

[21] Mettler B, Kong Z, Goerzen C, Whalley M. Benchmarking of obstacle field navigation algorithms for autonomous helicopters. In: Forum of the American Helicopter Society (AHS). Phoenix, AZ, USA; 2010. p. 1936-53.