Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Multi-Target Detection and Recognition by UAVs Using Online POMDPs

Caroline P. Carvalho Chanel[1,2] **and Florent Teichteil-Königsbuch**[2] **and Charles Lesire**[2]
name.surname@onera.fr
[1] Université de Toulouse – ISAE – Institut Supérieur de l'Aéronautique et de l'Espace
[2] Onera – The French Aerospace Lab, F-31055, Toulouse, France

Abstract

This paper tackles high-level decision-making techniques for robotic missions, which involve both active sensing and symbolic goal reaching, under uncertain probabilistic environments and strong time constraints. Our case study is a POMDP model of an online multi-target detection and recognition mission by an autonomous UAV. The POMDP model of the multi-target detection and recognition problem is generated online from a list of areas of interest, which are automatically extracted at the beginning of the flight from a coarse-grained high altitude observation of the scene. The POMDP observation model relies on a statistical abstraction of an image processing algorithm's output used to detect targets. As the POMDP problem cannot be known and thus optimized before the beginning of the flight, our main contribution is an "optimize-while-execute" algorithmic framework: it drives a POMDP sub-planner to optimize and execute the POMDP policy in parallel under action duration constraints. We present new results from real outdoor flights and SAIL simulations, which highlight both the benefits of using POMDPs in multi-target detection and recognition missions, and of our "optimize-while-execute" paradigm.

Introduction

Designing robots that are able to automatically plan for their long-term goals under uncertainty and partial observability of their environment, is challenging due to the various uncertain events to consider and to the "curse of dimensionality" issue of solving methods (Sabbadin, Lang, and Ravoanjanahary 2007; Ross and Chaib-Draa 2007; Smith and Simmons 2005; Bonet and Geffner 2009). High-level strategies, named policies, can be automatically generated and optimized using Partially Observable Markov Decision Processes (POMDPs) (Smallwood and Sondik 1973). Yet, current deployed systems either assume that the problem is available before the mission starts (Taha, Mir, and Dissanayake 2011; Spaan 2008; Bai et al. 2011; Schesvold et al. 2003), so that plenty of time is available to optimize off-line the strategy, or the problem is optimized on-line but only over a short-term horizon (Eidenberger, Grundmann, and Zoellner 2009; Miller, Harris, and Chong 2009).

In this paper, we address the problem of target detection and recognition by autonomous Unmanned Aerial Vehicles (UAVs), which is an active field of research (Wang et al. 2012). In such missions, the high-level decision strategy of UAVs (e.g. fly to a given zone, land, take image, etc.) depends on many stochastic events (e.g. target detected in a given zone, target recognized, etc.) that may arise when executing the decision rule. Because of the high complexity of solving POMDPs, few deployed UAV systems rely on automatically-constructed and long-term optimized policies.

Moreover, in a target detection and recognition mission, if viewed as an autonomous sequential decision problem under uncertainty, the problem is not known before the flight. Indeed, the number of targets and zones making up the environment are usually unknown beforehand: they must be automatically extracted at the beginning of the mission, in order to define the sequential decision problem to optimize. Therefore, contrary to most deployed or published approaches (see below), the POMDP model needed to optimize our UAV's policy is defined only during the flight, after the number of zones is analyzed online.

In order to tackle this challenging decision-theoretic robotic problem, we propose: (1) an *"optimize-while-execute"* paradigm to solve the POMDP online during the flight, taking into account time constraints required by the mission's duration and possible future execution states of the system; (2) an analysis of experimental results conducted on real flights. Our main contribution provides practical algorithmic means to optimize and execute in parallel an on-line POMDP policy under strong time constraints. Note that this contribution is different from the classical on-line approach used by POMDP algorithms like AEMS (Ross and Chaib-Draa 2007) or RTDP-bel (Bonet and Geffner 2009), which do not guarantee to provide optimized actions in constrained time when immediately required by the execution engine, which is essential for critical robotic missions. Indeed, previous approaches do not strongly control the time spent to optimize an action in the current or future execution states.

The paper is organized as follows. We discuss related work in the next section. We then present backgrounds on POMDP planning, and the POMDP model used to solve our target detection and recognition mission, as well as its automatic generation during the flight depending on the environment's topography. We also explain how we statistically build a probabilistic abstraction of the image processing algorithm's classifier output, which feeds our POMDP observation model. Then, we detail our "optimize-while-execute" algorithmic paradigm required to optimize and execute in parallel online POMDP policies. Finally, we discuss simulated and real-flight experiments.

Related work

POMDPs have been successfully implemented in many real robotic missions, e.g.: intelligent camera network for target tracking (Spaan 2008), ground robotics (Candido and Hutchinson 2011; Eidenberger, Grundmann, and Zoellner 2009), goal detection and active sensing in aerial robotic missions (Miller, Harris, and Chong 2009; Schesvold et al. 2003; Bai et al. 2011), and assistance to disabled persons (Taha, Mir, and Dissanayake 2011). From a practical viewpoint, these approaches can be globally classified in two groups. In the first group, the problem to solve is known in advance, so that the mission can be solved offline. Spaan (2008) uses a POMDP model to control a camera network in order to track targets. Bai et al. (2011) model an aircraft collision avoidance mission as a POMDP to automatically generate the threat resolution logic of the collision avoidance system. Offline POMDP methods have also been recently proposed to model the interaction between robotic devices and human operators (Taha, Mir, and Dissanayake 2011). In the second group, the POMDP problem is optimized online, but over a very short-term horizon – only the next action to perform in most approaches. Eidenberger, Grundmann, and Zoellner (2009) propose a POMDP model of active sensing with an optimization criterion based on information-theory. Miller, Harris, and Chong (2009) use a continuous POMDP framework for UAV's guidance in order to perform multi-target tracking. Finally, Candido and Hutchinson (2011) also tackle continuous POMDPs to address the problem of motion planning under collision avoidance constraints.

In this paper, due to the nature of our multi-target detection and identification mission in an uncertain environment, we need for a radically different approach. Namely, we have to optimize a complex POMDP problem known only at running time, which has to be solved over a long-term horizon in order to maximize information gathering and the chance of achieving a symbolic mission goal: landing near a specific target that has to be detected and distinguished from other targets in the scene.

Formal framework: POMDP

A POMDP is a tuple $\langle S, A, \Omega, T, O, R, b_0 \rangle$ where S is a set of states, A is a set of actions, Ω is a set of observations, $T : S \times A \times S \to [0; 1]$ is a transition function such that $T(s_{t+1}, a, s_t) = p(s_{t+1} \mid a, s_t)$, $O : \Omega \times S \to [0; 1]$ is an observation function such that $O(o_t, s_t) = p(o_t | s_t)$, $R : S \times A \to \mathbb{R}$ is a reward function associated with a state-action pair, and b_0 is the initial probability distribution over states. We note Δ the set of probability distributions over the states, called *belief state space*. At each time step t, the agent updates its *belief state* defined as an element $b_t \in \Delta$ using Bayes' rule (Smallwood and Sondik 1973).

Solving POMDPs consists in constructing a policy $\pi : \Delta \to A$, which maximizes some criterion generally based on accumulated rewards averaged over belief states. When symbolic rewarded goals must be achieved, like landing near a specific target, a criterion based on long-term average discounted accumulated rewards from any initial belief state is reasonable: $V^\pi(b) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(b_t, \pi(b_t)) | b_0 = b \right]$ where γ is the discount factor.

The optimal value of belief states is proven to be piecewise linear and convex, and solution of Bellman's equation (Smallwood and Sondik 1973): i.e, at step $n < \infty$, the value function can be represented by a set of hyperplanes over Δ, known as α-vectors. Actions are associated with α-vectors, and they are optimal over a region of the belief state space where their corresponding α-vectors dominate others.

Recent offline solving algorithms, e.g. PBVI (Pineau, Gordon, and Thrun 2003), HSVI2 (Smith and Simmons 2005) and SARSOP (Kurniawati, Hsu, and Lee 2008), and online algorithms like RTDP-bel (Bonet and Geffner 2009) and AEMS (Ross and Chaib-Draa 2007), approximate the value function with a bounded set of belief states B, where $B \subset \Delta$. These algorithms implement different heuristics to explore the belief state space using probabilistic trials, and update the value of V, which is represented by a set of α-vectors (except in RTDP-bel and AEMS), updating only the explored or relevant belief states $b \in B$. Therefore, V is reduced and contains a limited number $|B|$ of α-vectors.

Note that most – if not all – algorithms do *not* provide any means to control the time spent to update the value of a given belief state. They do not permit to stop at a *precise* time a belief set value update, or a trial in the belief space once launched. Thus, the execution engine must wait for the current value update over the set of beliefs to end, before applying the next action. For time-constrained robotic applications, time is so critical that we cannot assume that the time spent by the POMDP solver to optimize the next action is not strictly controllable. In this paper, we present a technique to leverage this cumbersome limitation of existing approaches.

POMDP model of the target detection and recognition mission

Mission description

We consider an autonomous Unmanned Aerial Vehicle (UAV) that must detect and recognize some targets under real-world constraints. The mission consists of detecting and identifying a car that has a particular model among several cars scattered in an unknown environment, and of landing close to this car. Due to the partial observability of objects in the scene, we model our mission as a POMDP with probabilistic beliefs about cars' models. The UAV can perform both high-level mission tasks and perception actions. We do not assume any prior number of cars in the scene, which can be in any of many zones in the environment (but no more than one car per zone). Zones are extracted online by image processing, before optimizing the POMDP policy.

The total number of states depends on variables that are all discretized: the number of zones (N_z), height levels (N_h), and car models (N_{models}), as well as a terminal state that characterizes the end of the mission. As cars (candidate targets) can be in any of the zones and be of any possible model a priori, and there is possibly no car in a given zone, thus the total number of states is: $|S| = N_z \cdot N_h \cdot (N_{models} + 1)^{N_z} + T_s$, where T_s represents the terminal state.

For this application case, we consider 5 possible observations ($|\Omega| = 5$) in each state: $\{$*no car detected, car detected but not identified, identified as model A, identified as model B, identified as model C*$\}$. These observations rely on the result of image processing (described later) and on the number of car models in the database.

High-level tasks performed by the autonomous UAV are: moving between zones, changing height level, changing view angle, landing. Actions for moving between zones (resp. height levels) are: $go\_to(\hat{z})$ (resp. $go\_to(\hat{h})$), where $\hat{z}$ (resp. $\hat{h}$) represents the zone (resp. height level) to go to. The *change_view* action changes the view angle when observing a given zone. The land action can be performed by the autonomous UAV at any moment and in any zone. Moreover, the land action finishes the mission. So, the total number of actions is: $|A| = N_z + N_h + 2$.

Model dynamics

State variables. The world state is described by discrete state variables. We assume that we have some basic prior knowledge about the environment: there are at most N_z zones, which contain each at most one target. The set of target models is finite, i.e. $N_{models} = \{model_A,\ model_B,\ model_C\}$. State variables are: z with N_z possible values, that indicates the UAV's position; h with N_h possible values, that indicates its height levels; $Id_{Ta_{z_1}}$ (resp. $Id_{Ta_{z_2}}$, $Id_{Ta_{z_3}}$, etc) with $N_{models} + 1$ possible values, that indicates the identity (car model) of target in z_1 (resp. target in z_2, etc.).

Transition and reward functions. Based on previous missions with our UAV, we know that moving and landing actions are sufficiently precise to be considered as deterministic: the effect of going to another zone, or changing flight altitude, or landing, is always deterministic. However, the problem is still a POMDP, because observations of cars' models are probabilistic. Moreover, it has been proved that the complexity of solving POMDPs essentially comes from probabilistic observations rather than from probabilistic action effects (Sabbadin, Lang, and Ravoanjanahary 2007).

In order to be compliant with the POMDP model, which assumes that observations are available after each action is executed, all actions of our model provide an observation of cars' models. The only possible observation after the landing action is *no car detected*. All other actions described bellow automatically take images of the scene available in front of the UAV, giving rise to image processing and classification of observation symbols (see later). As the camera is fixed, it is important to control the orientation of the UAV in order to observe different portions of the environment.

action $go\_to(\hat{z})$ ($go\_to(\hat{h})$)**:** this action brings the UAV to the desired zone (resp. height level). The cost associated with this action models the fuel consumption depending on the distance between zones (resp. difference between height levels) and the cost modeling the time spent by the image processing algorithm. Zone coordinates are only known at the beginning of the flight after a first exploration of the environment, so that the POMDP problem is automatically compiled during the flight given the extracted zones.

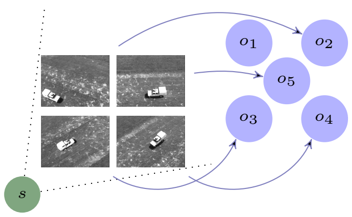
action $change\_view$**:** this is a high-level action that changes the view angle of the UAV when observing cars. Note that we do not assume any prior orientation of cars. This action does not change the POMDP state, which only relies on the UAV's current zone and height level. The execution engine is in charge of making *the UAV turn around the zone* of about 10 degrees (w.r.t the zone's center), in order to observe the target from a *different view angle*. This abstraction allows us to significantly reduce the number of state variables (hence POMDP states), while remaining realistic enough. The cost associated with the $change\_view$ action is proportional to the distance between the UAV position and the new position, and also to the computation time of the image processing engine, which is ran in each new view angle.

action $land$**:** this action completes the mission, leading the autonomous UAV to the terminal state. It brings a reward when the UAV lands in the zone where the searched target is actually located. Note that the model of the searched target (Se_{Ta}), i.e. one of $\{model_A,\ model_B,\ model_C\}$, is only known at the beginning of the mission during the flight.

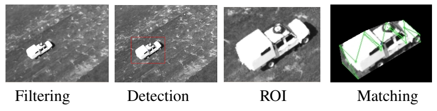
Finally, if the UAV is in the terminal state (T_s), all these actions have no effects and no cost.

Observation model. POMDP models require a proper probabilistic description of actions' effects and observations, which is difficult to obtain in practice for real complex applications. For our target detection and recognition mission, we automatically learned from real data the observation model, whose symbols are produced by an image processing algorithm. We recall that we consider 5 possible observations in each state: $\{$*no car detected, car detected but not identified, car identified as model A, car identified as model B, car identified as model C*$\}$. The key issue is to assign a prior probability on the possible semantic outputs of image processing given a particular scene.

Car observation is based on an object recognition algorithm (Saux and Sanfourche 2011), which is already embedded on-board our autonomous UAV. It takes as input one shot image (see Fig. 1(b)) that comes from the UAV onboard camera. First, the image is filtered to automatically detect if the target is in the image. If no target is detected, it directly returns the label *no car detected*. Otherwise, it extracts the region of interest of the image (bounding rectangle on the third picture in Fig. 1(b)), then generates a local projection and compares it with 3D template silhouettes recorded in a database of car models. The local projection only depends on the UAV's height level, camera focal length and azimuth as viewing-condition parameters. The current height level is updated at every time step, whereas the focal length and the camera azimuth are fixed parameters. Finally, the image processing algorithm chooses the 3D template that maximizes the similarity in the database, and returns the label that corresponds or not to the searched target: $model_A$, $model_B$ or



(a) More than 500 images per state.



(b) Target detection and recognition image processing (Saux and Sanfourche 2011).

obs. (o_i)	$\hat{p}(o_i\|s)$	$\sigma_{\hat{p}}$	$I = [\hat{p} \pm 3\sigma_{\hat{p}}]$
no car	0.0061	0.0023	[0 ; 0.0129]
not ident.	0.1177	0,0095	[0,0891 ; 0,1462]
as A	0,6809	0,0137	[0,6396 ; 0,7221]
as B	0,0601	0,0070	[0,0391 ; 0,0812]
as C	0,1351	0,0101	[0,1048 ; 0,1654]

(c) Exemple of probability table for a given state s; σ_p represents the standard error.

Figure 1: Identification of the observation model $\hat{p}(o|s)$, where $s = \{z, h, Id_{Ta_z} = model_A\}$.

$model_C$. If the level of similarity is less than a hand-tuned threshold, the image processing algorithm returns the label *car detected but not identified*.

In order to learn the POMDP observation model from real data, we performed many outdoor test campaigns with our UAV and some known cars. It led to an observation model learned via a statistical analysis of the image processing algorithm's answers, using images taken during these tests. More precisely, to approximate the observation function $O(o_t, s_t)$, we count the number of times that one of the 5 observations was an answer of the image processing algorithm in a given state s (see Fig. 1(a)). So, we compute the following estimation $\hat{p}(o_i|s)$, where o_i is one of the 5 possible observations: $\hat{p}(o_i|s) = \frac{1}{N_{exp}} \sum_{n=1}^{N_{exp}} \mathbb{I}_{\{o_n = o_i|s\}}$. N_{exp} represents the number of experiences, i.e. the number of runs of the image processing algorithm for the different images, and o_n the label obtained at experience n. It applies the usual estimator of the mean of a Bernoulli distribution, which estimates the probability of o_i against the other observations. It is proven to converge in probability to $p(o_i|s)$ as $N_{exp} \to \infty$. More than 500 images are analyzed for each state ($N_{exp} \gg 1$), so that the statistical approximations may be good enough. We also computed intervals in which the true model lies with 99% confidence.

Moreover, this learning method can be performed off-line because it is independent from the zones that will be extracted during the mission. The estimation $\hat{p}(o_i|s)$ is computed for each possible target and height level, but independently from the zone used to learn it. Finally, this parametric model allows us to automatically generate the POMDP model at the beginning of the mission, after an initial flight phase where zones of interest are detected by a specific image processing algorithm not described in this paper. Thereby, once zones are discovered, the POMDP model's rewards can be generated on the basis of zone coordinates.

Fig. 1(c)'s table shows an example of learned observation probabilities. We precomputed such tables for all possible state variable instantiations on which the observations depend. Note that no learning phase to further improve the observation model is performed during the actual mission flight. We dismissed the use of reinforcement learning techniques because: (1) we need to learn only the observation model, but not the actions' effects nor the reward function; and (2) our UAV is too expensive to take the risk to destroy it while learning the best strategy.

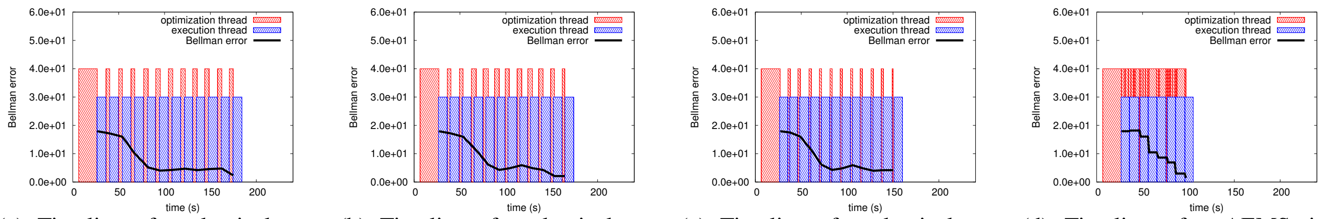
Parallel optimization and execution

Complex UAV missions defined as large POMDPs can rarely be optimized online, because of the lack of sufficient computational means. Moreover, the problem to solve is not always known in advance, e.g. our target detection and recognition mission, where the POMDP problem is based on zones that are automatically extracted from online images of the environment. Such applications require an efficient online framework for solving large POMDPs under strong time constraints, and executing policies before the mission's deadline. We worked on extending the *"optimize-while-execute"* framework proposed in (Teichteil-Konigsbuch, Lesire, and Infantes 2011), restricted to deterministic or MDP planning, to POMDPs. Concretely, we generalized this algorithmic framework, which was based on completely observable states, to belief states and point-based paradigms used by state-of-the-art POMDP planners.

The extension is not an easy task. In (Teichteil-Konigsbuch, Lesire, and Infantes 2011), the execution engine and the planner have full access to the system's states. In a POMDP context, the execution engine and the planner can only access observations, so that they have to maintain a belief state (probability distribution over states). The idea is to drive a standard POMDP planner which is called from possible future beliefs states while executing the current optimized action in the current belief state, in anticipation of the probabilistic evolution of the system-environment pair.

The *optimize-while-execute* paradigm is depicted in Alg. 1. It is an anytime meta planner, composed of an execution component and a planning component. The latter drives any POMDP planner like PBVI, HSVI, AEMS, RTDP-bel. It is strictly *anytime* in the sense of policy execution under time constraints: the planning component ensures to return an applicable action in any possible belief state at a *precise* time point, when required by the execution component.

First, let us define a planning request: it consists of a request sent by the execution engine to the meta planner, containing: a future possible belief state from which the planner must compute an optimized action, the time allocated to improve the policy from this belief state, the algorithm used to improve the policy, and its parameters. Second, an action request is used by the execution engine to get the action whose optimization was launched at some time in the past for the current belief state. If no action is available, a heuristic default action is returned. Each component works on an independent thread: the execution engine, which manages policy execution, runs in the execution thread that interacts with the system's engine; it competes with the meta planner, which is



(a) Timelines for classical use of AEMS (interleaved approach) with 4s for planning.

(b) Timelines for classical use of AEMS (interleaved approach) with 3s for planning.

(c) Timelines for classical use of AEMS (interleaved approach) with 2s for planning.

(d) Timelines for AEMS implementation in our *"optimize-while-execute"* framework.

Figure 2: Interleaving optimization and execution versus *optimize-while-execute* paradigm.

Algorithm 1: `optimize-while-execute`

```
1  $b \leftarrow$ initial belief state;
2  Send planning_request($b$, bootstrap time, algorithm) ;
3  Wait for bootstrap's end;
4  while stop meta planner requested $=$ false do
5      $a \leftarrow$ action_request($b$);
6      Execute action $a$;
7      $\Delta_a \leftarrow$ expected duration of action $a$;
8      $O \leftarrow$ get effects of action $a$ (possible observations);
9      for $0 < i < |O|$ do
10         $b' \leftarrow$ compute next belief from Bayes' rule ;
11         $\Delta' \sim \Delta_a * p(o|a, b)$;
12         Send planning_request($b'$, $\Delta'$, algorithm);
13     Wait until action $a$ has completed;
14     $o \leftarrow$ get observation;
15     Remove all planning requests;
16     $b \leftarrow$ update current belief state with $a$ and $o$;
```

in charge of policy optimization in a concurrent thread.

Initially, the execution engine sends a planning request to the meta planner for the initial belief state (bootstrap, line 2). After a certain amount of time (line 3), the execution engine sends an action request in order to get the optimized action for this belief (line 5). Then, while executing the action optimized in the current belief state, it will anticipate the environment's evolution by planning in advance for the next possible belief states.

To compute next belief states, we ask the anytime planner what are the probabilistic effects of the action that is being run in the current belief state. We consider observations as effects, so we construct the next belief states for each possible observation (lines 8-10). And so, we compute the policy for each of them during a time proportional to the action's predicted duration and to the probability of each possible observation[1] (line 11).

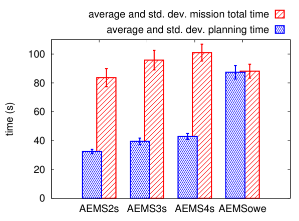
We implemented this meta planner using the state-of-the-art algorithm AEMS as the underlying POMDP planner. AEMS is particularly useful for our *optimize-while-execute* framework, since it is more reactive than other online POMDP planners. Each iteration of AEMS' main loop is usually faster than other solving methods (Ross and Chaib-Draa 2007), which increases the chance that an action has been optimized in the belief state of an incoming action request. It also allows our framework to update more often the belief states of the queued optimization requests during their allocated times, thus providing better optimized actions than using other underlying POMDP planners. Note that AEMS actually relies on a global allocated planning time, but we still cannot control the time spent in each trial, thus the need for our framework.

Why optimizing-while-executing?

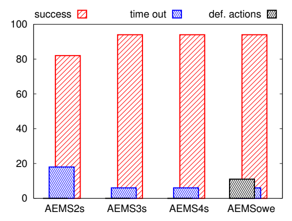
Figure 2 highlights the benefits of our approach compared with the classical approach of AEMS for different planning times (4, 3 and 2 seconds), which consists in interleaving planning and execution, i.e. it plans for the current belief state (for a long-term horizon) at every decision epoch, but not in advance for the future ones as in our *optimize-while-execute* approach. The x axis represents the time elapsed during the mission. Red (resp. blue) bars represent the portions of time where the policy is optimized in the optimization thread (resp. executed in the execution thread) – both running in parallel when solving our target detection and recognition POMDP. The curve shows the evolution of Bellman's error during the mission, which emphasizes the evolution of the optimization process. To highlight our approach, we fixed the mission's maximum allowed duration to 180 seconds. With the classical use of AEMS (Fig. 2(a), 2(b) and 2(c)), we can easily notice that the mission's total time increases with the time allocated to plan from the current execution state. Successive red bars show that the POMDP needs to be (re-)optimized in each new execution state. On the contrary, our approach (Fig. 2(d)) permanently optimizes for future possible execution states while executing the action in the current execution state, so that optimized actions are always *immediately* available in each new execution state. As the result, the mission's duration is lower with our approach than with the interleaved approach (at least 30% less).In other words, in our approach the amount of time saved relies on the sum of time slices of the classical approach when the optimization thread is idle. The more actions get time to be executed, the more time will be saved.

In Fig. 3, we still compare our *optimize-while-execute* approach using AEMS, named AEMSowe in the plots, with the traditional interleaved use of AEMS for different planning times. This comparison is made on two study cases with 3 zones ($N_z = 3$). In case 1, the searched target is located in zone 2, while in case 2 it is in zone 3. The UAV starts in zone 1. For each environmental setting and each algorithmic version of AEMS, we ran 50 software-architecture-in-the-loop (SAIL) simulations of the mission using images taken

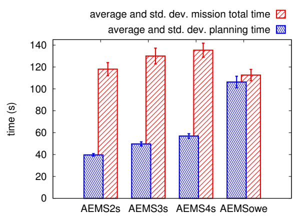
[1]We actually defined several time allocation strategies, like giving priority to belief states whose estimated value is more uncertain, but we have not evaluated and compared them for the moment.



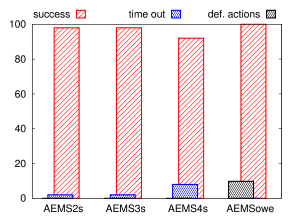
(a) Avg. time for successful missions for the study case 1.



(b) Percentages among missions for the study case 1.



(c) Avg. time for successful missions for the study case 2.



(d) Percentages among missions for the study case 2.

Figure 3: Results for software-in-the-loop simulations.

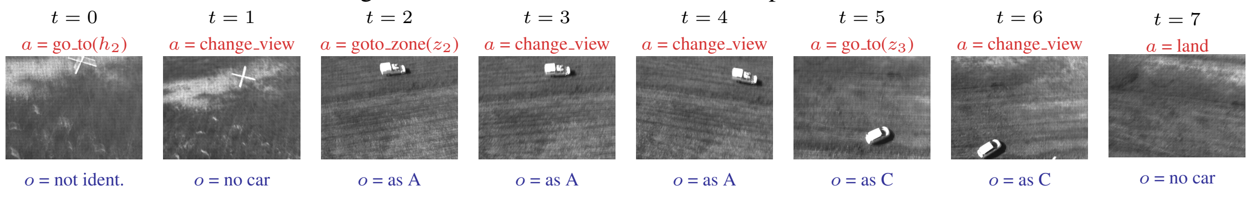


Figure 4: Sequence of decisions for time step t. Each image represents the input of the image processing algorithm after the current action a is executed. Observations o represent the successive outputs of the image processing algorithm's classifier.

during real flights. Our SAIL simulations use the exact functional architecture and algorithms used on-board our UAV, a Yamaha Rmax adapted to autonomous flights, as well as real outdoor images. We averaged the results and analyzed the total mission time and planning time, the percentage of timeouts and successes in terms of landing near the searched car. Action durations are uniformally drawn in $[T^a_{min}, T^a_{max}]$, with $T^a_{min} = 8s$ and $T^a_{max} = 10s$, which is representative of durations observed during real test flights. As expected, AEMSowe permanently optimizes the policy in background, contrary to the interleaved approach. As a result, it is more reactive: it has the minimum mission's time (Fig. 3(a) and 3(c)), while providing the best percentage of successes and the minimum number of timeouts (Fig. 3(b) and 3(d)). Note that, in Fig 3(a), AEMS2s performs better in averaged mission time (avg. over successful missions), but the percentage of successful missions is lower than in our approach ((Fig. 3(b)). Furthermore, less than 20% of default actions were used, which shows the relevance of optimizing actions in advance for the future possible belief states.

Real flight experiments

In this section, we present results obtained during real flight experiments, where the *optimize-while-execute* framework was embedded on-board the UAV. We considered 2 height levels (30 and 40 meters), at most 3 targets of 3 possible car models, and 3 zones are extracted at the beginning of the mission. The aim is to land next to the car model C; however, the number of cars actually present in the scene and the models of these cars are unknown at the beginning of the mission. These real flights involve few states, but note that the need for POMDPs in this kind of applications is more related to the expressiveness and accuracy of probabilistic observation functions, than to the size of the problem.

The trajectory performed by the UAV is shown in Fig. 5. The coordinates are expressed in the local reference system of the experimental flight terrain. The black cross, orange cross and blue star, respectively represent zone 1, 2 and 3 (where the searched target is located).

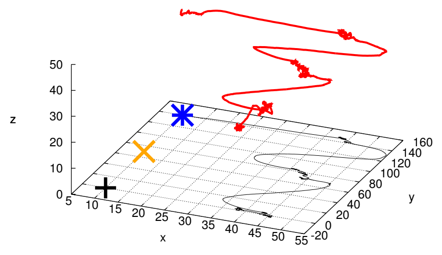


Figure 5: UAV's trajectory performed during the real flight.

Figure 4 presents the sequence of actions performed by the autonomous UAV per time step during one of the real flights. Initially, the UAV is in zone 1 at an altitude of 30 meters. After the 2 first steps, the UAV observes that there is no car in this zone, thus it chooses to change to zone 2. After 3 steps, it concludes that the target observed in zone 2 is of type A. So, it decides to change to zone 3, and after 2 more observations, the UAV believes that the target C is in zone 3 (which is correct) and chooses to land. In this way it correctly terminates the mission.

Conclusion and future work

To the best of our knowledge, this paper presents one of the first POMDP real flight implementations of multi-target detection and recognition mission by an autonomous UAV using AI modern planning techniques. The POMDP model is automatically compiled during the flight and relies on a meaningful probabilistic observation model learned from the outputs of an image processing algorithm. The POMDP optimized policy deals both with sensing actions (determining the best view angle for each target), and with symbolic goals (landing near the searched target). We also provide practical algorithmic means to optimize and execute POMDP policies in parallel under time constraints, which is required because the POMDP problem is unknown before the flight and the mission's duration is especially short.

Our experiments, particularly real flights, demonstrate that POMDP planning techniques are now mature enough to tackle complex aerial robotic missions, assuming the use of realistic observation models and some kind of *"optimize-while-execute"* framework, as proposed in this paper. Possible future research improvements include: taking into account safety constraints imposed by civil aeronautical agencies when optimizing the strategy; and building POMDP policies that deal with the standard deviations or the confidence intervals of the learned observation model.

References

Bai, H.; Hsu, D.; Kochenderfer, M.; and Lee, W. S. 2011. Unmanned Aircraft Collision Avoidance using Continuous-State POMDPs. In *Robotics Science and Systems (RSS'11)*.

Bonet, B., and Geffner, H. 2009. Solving POMDPs: RTDP-bel vs. point-based algorithms. In *Int. Joint Conf. on Artificial Intelligence (IJCAI'09)*.

Candido, S., and Hutchinson, S. 2011. Minimum uncertainty robot navigation using information-guided POMDP planning. In *Int. Conf. on Robotics and Automation (ICRA'11)*.

Eidenberger, R.; Grundmann, T.; and Zoellner, R. 2009. Probabilistic action planning for active scene modeling in continuous high-dimensional domains. In *Int. Conf. on Robotics and Automation (ICRA'09)*. IEEE.

Kurniawati, H.; Hsu, D.; and Lee, W. 2008. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics Science and Systems (RSS'08)*.

Miller, S. A.; Harris, Z. A.; and Chong, E. K. P. 2009. A POMDP framework for coordinated guidance of autonomous UAVs for multitarget tracking. *EURASIP J. Adv. Signal Process* 2:1–2:17.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*.

Ross, S., and Chaib-Draa, B. 2007. AEMS: An anytime online search algorithm for approximate policy refinement in large POMDPs. In *Int. Joint Conf. on Artificial Intelligence (IJCAI'07)*.

Sabbadin, R.; Lang, J.; and Ravoanjanahary, N. 2007. Purely epistemic Markov Decision Processes. In *AAAI-07*.

Saux, B., and Sanfourche, M. 2011. Robust vehicle categorization from aerial images by 3D-template matching and multiple classifier system. In *Int. Symposium on Image and Signal Processing and Analysis (ISPA)*.

Schesvold, D.; Tang, J.; Ahmed, B.; Altenburg, K.; and Nygard, K. 2003. POMDP planning for high level UAV decisions: Search vs. strike. In *Int. Conf. on Computer Applications in Industry and Enineering*.

Smallwood, R., and Sondik, E. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 1071–1088.

Smith, T., and Simmons, R. 2005. Point-based POMDP algorithms: Improved analysis and implementation. In *Conf. on Uncertainty in Artificial Intelligence (UAI'05)*.

Spaan, M. 2008. Cooperative Active Perception using POMDPs. *Association for the Advancement of Artificial Intelligence - AAAI*.

Taha, T.; Mir, J. V.; and Dissanayake, G. 2011. A pomdp framework for modelling human interaction with assistive robots. In *Int. Conf. on Robotics and Automation (ICRA'11)*.

Teichteil-Konigsbuch, F.; Lesire, C.; and Infantes, G. 2011. A generic framework for anytime execution-driven planning in robotics. In *Int. Conf. on Robotics and Automation (ICRA'11)*.

Wang, J.; Zhang, Y.; Lu, J.; and Xu, W. 2012. A Framework for Moving Target Detection, Recognition and Tracking in UAV Videos. In *Advances in Intelligent and Soft Computing*, volume 137. Springer Berlin / Heidelberg. 69–76.