Temporal Logic Motion Planning using POMDPs with Parity Objectives*

Case Study Paper

Mária Svoreňová
Faculty of Informatics
Masaryk University
Brno, Czech republic
svorenova@mail.muni.cz

Martin Chmelík
IST Austria
Klosterneuburg, Austria
mchmelik@ist.ac.at

Kevin Leahy
Dep. of Mech. Engineering
Boston University
Boston, MA, USA
kjleahy@bu.edu

Hasan Ferit Eniser
Bogazici University
Istanbul, Turkey
hasan.eniser@boun.edu.tr

Krishnendu Chatterjee
IST Austria
Klosterneuburg, Austria
kchatterjee@ist.ac.at

Ivana Černá
Faculty of Informatics
Masaryk University
Brno, Czech republic
cerna@muni.cz

Calin Belta
Dep. of Mech. Engineering
Boston University
Boston, MA, USA
cbelta@bu.edu

ABSTRACT

We consider a case study of the problem of deploying an autonomous air vehicle in a partially observable, dynamic, indoor environment from a specification given as a linear temporal logic (LTL) formula over regions of interest. We model the motion and sensing capabilities of the vehicle as a partially observable Markov decision process (POMDP). We adapt recent results for solving POMDPs with parity objectives to generate a control policy. We also extend the existing framework with a policy minimization technique to obtain a better implementable policy, while preserving its correctness. The proposed techniques are illustrated in an experimental setup involving an autonomous quadrotor performing surveillance in a dynamic environment.

1. INTRODUCTION

Accounting for uncertainty is a challenging problem in robot motion planning [19]. Partially observable Markov decision processes (POMDPs) [11] are widespread models that capture the uncertainties inherent in a robot's actuators and sensors. While solving large POMDPs is a particularly difficult problem, significant progress has been made recently in computing approximate solutions using point-based algorithms [12, 18, 17, 10, 16]. Such techniques have been successfully applied to several moderately complex robotic tasks, including navigation [14, 3], grasping [7], target tracking [12, 8], and exploration [17]. In some cases, POMDPs with hundreds of states have been solved in a matter of seconds (see, e.g., [17, 8]). In these works, the mission usually requires to move a robot from an initial to a final location in finite time, while minimizing a cost. Several applications, however, require the accomplishment of more complex missions, possibly over infinite time horizons. For example, in a persistent surveillance mission, an autonomous aircraft might be required to "keep on collecting data from region A and uploading it in region B while always avoiding region C.". Such specifications translate naturally to formulas of temporal logics, such as Linear Temporal Logic [6].

In this work, we focus on the case study of the problem of deploying an autonomous air vehicle in a partially observable, dynamic, indoor environment from a specification given as a linear temporal logic (LTL) formula. Every temporal logic property can be translated to a deterministic parity automaton [15, 13] that defines a parity objective on the POMDP. POMDPs with parity objectives are, therefore, quite general models for motion planning. We consider the *qualitative analysis* problem, i.e., the problem whether there exists a control policy that satisfies the property with probability 1 (almost-surely). The almost-sure satisfaction of the property provides the strongest probabilistic guarantee, which is also robust with respect to modeling errors.

The almost-sure analysis of POMDPs with parity objectives was shown to be undecidable in [1]. Recently, in [5], it was shown that when restricted to the practical case of finite-memory policies, the problem becomes decidable. In [4], the authors introduce a number of heuristics to deal with the exponential complexity of the problem and implement an algorithm for qualitative analysis of POMDPs under finite-memory policies. Whenever there exists an almost-sure winning policy, the algorithm also outputs a witness policy.

In this paper, we build on the results from [5, 4] to develop efficient algorithms to solve POMDPs corresponding to a class of robotic applications. The contribution of this case-study paper is threefold. First, we show how POMDP models can be constructed for a class of experimental robotic problems. Second, we show that policies constructed using [5, 4] can be minimized to obtain smaller, possibly easier to implement policies, while preserving their correctness. Finally, the proposed techniques are illustrated using computer simulation as well as experiments using a robotic testbed.

The rest of the paper is organized as follows. We describe the case study in Sec. 2. In Sec. 3, we introduce POMDPs with parity objectives, summarize the existing technical ap-

*The research was partially supported by Austrian Science Fund (FWF) Grant No P 23499-N23, FWF NFN Grant No S11407-N23 (RiSE), ERC Start grant (279307: Graph Games), Czech Ministry of Education Youth and Sports grant LH11065, NSF NRI-1426907, and ONR MURI N00014-10-10952.

HSCC '15 April 14 - 16, 2015, Seattle, WA, USA

http://dx.doi.org/10.1145/2728606.2728617.

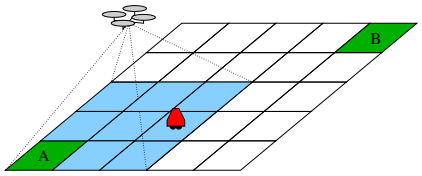

Figure 1: Case study of a quadrotor and a ground agent moving in a shared partitioned environment. The quadrotor is equipped with a camera with a restricted field of view, for the current region $r_{2,2}$ shown in blue. The quadrotor's objective is to survey regions $A = r_{1,1}$ and $B = r_{5,5}$, in green, while avoiding the ground agent, currently in region $r_{2,3}$.

proach and present our new technical result of policy minimization. Finally, in Sec. 4, we build the POMDP model for the case study, synthesize corresponding policies, and demonstrate their performance using simulations and experiments on a real platform.

2. PROBLEM FORMULATION

We consider a quadrotor and a ground agent moving synchronously and independently in a shared environment. The environment is discretized into a grid of m by n equally sized square regions (a 5 by 5 grid is depicted in Fig. 1). The objective for the quadrotor is to survey regions A and B, shown in green in Fig. 1, while avoiding the ground agent, *i.e.,* never enter a configuration, where the quadrotor hovers above the ground agent in the same region of the grid. The quadrotor is equipped with a downward facing camera with a restricted field of view (FOV) of u by v regions, *e.g.,* the case of 3 by 3 field of view in quadrotor's current position in Fig. 1 is shown in blue. We consider three scenarios that differ in the image processing for the camera output and assumptions on the ground agent's motion. We synthesize a (finite-memory) policy for the quadrotor that satisfies the objective with probability 1 (almost-surely).

In this section, we describe the components of the setup and summarize the scenarios. We consider a 5 by 5 grid with a 3 by 3 FOV for the camera, as in Fig. 1. To illustrate scalability, we discuss larger environments in Sec. 4.

2.1 Components of the model

Environment. The environment is discretized into a 5 by 5 grid, where regions in the grid are $\mathcal{R} = \{r_{i,j} \mid 1 \leq i, j \leq 5\}$, i.e., $r_{i,j}$ refers to the region in the i-th row and j-th column ($r_{1,1}$ is the bottom left corner) of the grid as in Fig. 1.

Quadrotor. The quadrotor can move in the grid from its current region to any adjacent region in the grid or hover over the current region. The motion of the quadrotor is modeled by a set of actions $A = \{\mathsf{N}, \mathsf{S}, \mathsf{E}, \mathsf{W}, \mathsf{X}\}$. The first four actions correspond to the movement in the four compass directions and action X corresponds to hovering over the current region. The effects of all the actions are deterministic and correspond to the movement in the desired direction, *i.e.,* applying action E in region $r_{1,1}$ moves the quadrotor to region $r_{1,2}$. We assume that the quadrotor is initially positioned over region $r_{1,1}$.

Ground agent. The ground agent moves probabilistically in the grid. Just like the quadrotor, the ground agent is capable of moving from its current region to neighboring ones or to remain in the same region. We consider two particular motion models for this agent.

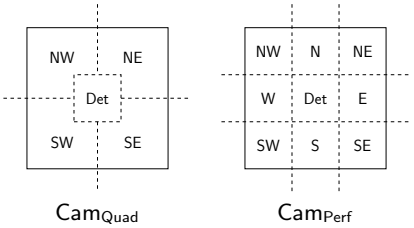

Figure 2: The two camera models and the names of observations transmitted from the camera to the quadrotor, if the ground agent is present in given parts of the FOV. In $\mathsf{Cam}_{\mathsf{Quad}}$, the captured image is divided into quadrants. In $\mathsf{Cam}_{\mathsf{Perf}}$, the image is divided into a grid of 9 cells.

Unrestricted ground agent. In the first, unrestricted motion model GA_u, we assume that the agent moves randomly across the whole grid, *i.e.,* the ground agent is initially positioned uniformly at random among the regions of the grid that are out of the quadrotor's FOV. The next visited region is sampled at random uniformly over the possible next regions. For example, from region $r_{2,1}$, the agent moves to any of the regions $r_{3,1}, r_{2,2}, r_{1,1}, r_{2,1}$ with probability $1/4$.

Restricted ground agent. In the second motion model GA_r, we assume that the ground agent moves randomly in the grid but can never enter the corner regions of the grid, *i.e.,* regions $r_{1,1}, r_{5,1}, r_{5,5}, r_{1,5}$. As in the first case, the ground agent is initially positioned in a region chosen uniformly at random from regions outside of the quadrotors' FOV that are not corner regions. The transition probability distribution in every region is the uniform distribution over the possible next regions, *e.g.,* from region $r_{2,1}$, the agent moves to any of the regions $r_{3,1}, r_{2,2}, r_{2,1}$ with probability $1/3$.

Camera. The downward facing camera mounted on the quadrotor provides feedback about the relative position of the ground agent. We consider two different image processing mechanisms for the camera.

Quadrant camera. In the first case $\mathsf{Cam}_{\mathsf{Quad}}$, the image captured by the camera is divided into quadrants as depicted in Fig. 2, *i.e.,* the camera is not able to determine in which region of the grid the ground agent currently is, it only determines its relative position to the current position of the quadrotor. The set of observations transmitted to the quadrotor is $Obs_{\mathsf{Quad}} = \{\mathsf{SW}, \mathsf{NW}, \mathsf{NE}, \mathsf{SE}, \mathsf{Det}, \mathsf{None}\}$, where the first four observations correspond to the ground agent being in the respective quadrant, Det stands for the agent being directly below the quadrotor, and None refers to the agent being out of the FOV. If the ground agent is out of FOV, the observation is None with probability 1. If the agent is in the FOV and in a region that intersects only one of the four quadrants, the observation is the corresponding quadrant with probability 1, *e.g.,* if the quadrotor is in region $r_{1,1}$ and the ground agent in $r_{2,2}$ the observation transmitted to the quadrotor is NE with probability 1. Finally, if the agent is in the FOV and in a region that intersects two of the four quadrants, the camera can report as the current observation any of the two quadrants, each with probability $1/2$, *e.g.,* if the quadrotor is in region $r_{1,1}$ and the ground agent in $r_{2,1}$ the observation transmitted to the quadrotor can be NW or NE each with probability $1/2$.

Perfect camera. The second image processing mechanism $\mathsf{Cam}_{\mathsf{Perf}}$ divides the captured image into a grid of 9 equally sized cells as shown in Fig. 2. Together with the fact that the image captures up to 9 regions of the

grid, this version of the camera allows for perfect recognition of the ground agent's position in FOV. The set of observations transmitted to the quadrotor is $Obs_{\mathsf{Perf}} = \{\mathsf{S}, \mathsf{N}, \mathsf{W}, \mathsf{E}, \mathsf{SW}, \mathsf{NW}, \mathsf{NE}, \mathsf{SE}, \mathsf{Det}, \mathsf{None}\}$ that correspond to the relative position of the ground agent. The position of the ground agent is detected correctly with probability 1, *e.g.,* if the quadrotor is in region $r_{1,1}$ and the ground agent in $r_{2,1}$, the observation is N with probability 1. The observations Det and None have the same meaning as in the case of the $\mathsf{Cam}_{\mathsf{Quad}}$ camera. For example, if the quadrotor is in region $r_{1,1}$ and the ground agent in $r_{2,1}$ the observation transmitted to the quadrotor is N with probability 1.

Scenarios. In this work, we consider the following three scenarios that arise from combinations of the above models.

Scenario 1
Considers the unrestricted motion model GA_u for the ground agent and the quadrant camera model $\mathsf{Cam}_{\mathsf{Quad}}$.

Scenario 2
Replaces the ground agent motion model with the restricted variant GA_r, and consider the quadrant camera $\mathsf{Cam}_{\mathsf{Quad}}$.

Scenario 3
Considers the unrestricted motion model GA_u for the ground agent and the perfect camera $\mathsf{Cam}_{\mathsf{Perf}}$.

3. TECHNICAL APPROACH

In this section we introduce POMDPs and describe how a finite-state policy can be synthesized by the algorithm presented in [4]. We also present a new technical contribution, which is a new approach to minimize finite-memory almost-sure winning policies in POMDPs with parity objectives.

3.1 POMDPs and Parity Objectives

A probability distribution f on a finite set X is a function $f : X \to [0,1]$ such that $\sum_{x \in X} f(x) = 1$, and we denote by $\mathcal{D}(X)$ the set of all probability distributions on X. For $f \in \mathcal{D}(X)$, we use $\mathrm{Supp}(f) = \{x \in X \mid f(x) > 0\}$ to denote the support of f. For probability distributions f and f' we will write $f \cong f'$ iff $\mathrm{Supp}(f) = \mathrm{Supp}(f')$.

POMDPs. A discrete-time *partially observable Markov decision process (POMDP)* is a tuple $\mathcal{M} = (S, A, \delta, Obs, \Pi, \mu)$, where (i) S is a finite set of *states*; (ii) A is a finite alphabet of *actions*; (iii) $\delta : S \times A \to \mathcal{D}(S)$ is a *probabilistic transition function* that given a state s and an action $a \in A$ gives the probability distribution over the successor states, *i.e.,* $\delta(s,a)(s')$ denotes the transition probability from state s to state s' given action a; (iv) Obs is a finite set of *observations*; (v) $\Pi : S \to \mathcal{D}(Obs)$ is a *probabilistic observation function* that maps every state to a distribution over the observations; and (vi) μ is the initial state distribution. We assume there is a finite set of atomic propositions AP, and a function $L : S \to 2^{AP}$ that labels a state s of the POMDP with a set of atomic proposition true at state s.

Plays and belief-supports. A *play* in a POMDP is an infinite sequence of states and actions $(s_0, a_0, s_1, a_1, \ldots)$ such that for all $i \geq 0$, we have $\delta(s_i, a_i)(s_{i+1}) > 0$. We write Ω for the set of all plays. For a finite prefix $w \in (S \cdot A)^* \cdot S$ of a play, we use $\mathsf{Last}(w)$ to denote the last state of w. For a finite prefix $w = (s_0, a_0, s_1, a_1, \ldots, s_n)$, we denote by $\Pi(w) = (\Pi(s_0), a_0, \Pi(s_1), a_1, \ldots, \Pi(s_n))$ the observation and action sequence associated with w. For a finite sequence $\rho = (z_0, a_0, z_1, a_1, \ldots, z_n)$ of observations and actions, the *belief-support* $\mathcal{B}(\rho)$ after the prefix ρ is the set of states in which a finite prefix of a play is with positive probability after the sequence ρ of observations and actions, *i.e.,* $\mathcal{B}(\rho) = \{s_n = \mathsf{Last}(w) \mid w = (s_0, a_0, s_1, a_1, \ldots, s_n), w \text{ is a prefix of a play, and for all } 0 \leq i \leq n.\ \Pi(s_i) = z_i\}$.

Policies with memory and finite-memory policies. A *policy* with memory is a tuple $\sigma = (\sigma_u, \sigma_n, M, m_0)$, where (i) M is a denumerable set (finite or infinite) of memory elements (or memory states); (ii) $\sigma_n : M \to \mathcal{D}(A)$ is the *next action selection function* that given the current memory state gives the probability distribution over actions; (iii) $\sigma_u : M \times Obs \times A \to \mathcal{D}(M)$ is the *memory update function* that given the current memory state, the current observation and action, updates the memory state probabilistically; and (iv) $m_0 \in M$ is the initial memory state. A policy is a *finite-memory* policy if the set M of memory elements is finite. A policy is *memoryless* if the set of memory elements M contains a single memory element.

Objectives. An *objective* specifies the desired set $\varphi \subseteq \Omega$ of plays (or behaviors) in a POMDP. A common approach to specify objectives is using LTL formulas [6]. They can express all commonly used specifications in practice in a way that resembles natural language. We use the graphical notation for LTL temporal operators, *i.e.,* eventually ($\Diamond$), always ($\Box$), next ($\bigcirc$) and until ($\mathcal{U}$).

In this work, we consider POMDPs with *parity objectives*, since every LTL formula can be translated to a deterministic parity automaton [15, 13]. Given a POMDP, an LTL formula, and an equivalent deterministic parity automaton for the formula, the synchronous product of the POMDP and the automaton is a POMDP with a parity objective. Formally, a parity objective φ is given by a priority function $p_\varphi : S \to \mathbb{N}$ that associates every state of the POMDP with a non-negative priority. A play $\rho = (s_0, a_0, s_1, a_1, s_2, a_2, \ldots)$ is then called winning with respect to the given parity objective if the minimum priority appearing infinitely often in the play is even.

Qualitative analysis. Given a policy σ for a POMDP, let $\mathbb{P}^\sigma_{s_0}(\cdot)$ denote the unique probability measure obtained by fixing the policy in the POMDP [20]. A policy σ is *almost-sure winning* for a parity objective φ, if $\mathbb{P}^\sigma_{s_0}(\varphi) = 1$. The *qualitative* analysis problem given a POMDP and a parity objective asks for an almost-sure winning policy.

3.2 Policy Synthesis and Minimization.

Policy synthesis. The algorithm presented in [4] decides whether there exists a finite-memory policy that satisfies a given a parity objective with probability 1. The algorithm first reduces the input POMDP to a polynomially larger POMDP with a parity objective with only two priorities. In the next step an exponential POMDP is constructed in which memoryless policies are sufficient. Finally, a memoryless policy in the final POMDP is translated back to a finite-memory policy in the input POMDP. The main purpose of the algorithm from [4] is to decide whether there exists an almost-sure winning policy. As a side effect of the computation it outputs an almost-sure winning policy $\sigma = (\sigma_u, \sigma_n, M, m_0)$. However, the policy can contain many redundant memory elements.

Policy minimization. One important aspect for efficient policies is to minimize the policy to obtain smaller policies. In this paper, we present a new approach to minimize policies that takes advantage of the following property of winning policies on POMDPs. Let $\sigma = (\sigma_u, \sigma_n, M, m_0)$ be a finite-memory almost-sure winning policy. It holds that by replacing any distribution f assigned by one of the functions

σ_u or σ_n by a different distribution f' such that $f \cong f'$, the new modified policy remains almost-sure winning [5]. In other words, policy σ is robust with respect to perturbations of its probability distributions. It follows, that one can view the policy σ as a nondeterministic finite-state automaton (NFA), where the states are the memory elements M, the initial state is m_0, the alphabet is the product $Obs \times A$, and the transition relation from state m and action (z, a) is $\delta(m, (z, a)) = \text{Supp}(\sigma_u(m, z, a))$. As NFA minimization is PSPACE-complete [9], the standard approach to obtain smaller NFAs is to compute a quotient NFA [2], where two states m and m' belong to the same quotient if the two states are bisimilar, i.e., $m \sim m'$. As there are no final memory elements in policies, we need to define a new initial equivalence relation $\sim_0$ on the states, that is then further refined. Every memory element of the policy σ returned by the algorithm contains as one of its components the current belief-support. For memory elements m and m', we define $m \sim_0 m'$ if the memory elements agree on the belief-support component. Using the standard algorithm for the computation of the bisimulation equivalence, the initial relation $\sim_0$ is refined until a fixpoint $\sim$ is reached, as in the case of the algorithm computing the quotient NFA [2]. The quotient NFA is translated back to an almost-sure winning policy.

4. RESULTS

In this section, we show how the considered scenarios can be modeled with POMDPs and formalize the quadrotor's objective using an LTL formula. In the next step, we apply the tool introduced in [4] with the policy minimization technique designed in Sec. 3.2 to solve the POMDPs. The constructed policies are intuitively interpreted over the corresponding scenarios. We also discuss the scalability of the approach over larger grids. Finally, we analyze and demonstrate the obtained policies using simulation in Matlab and in a robotic testbed.

4.1 POMDP models

Given a scenario, we construct a POMDP $\mathcal{M}$ that captures the interaction of all components. Here we describe the POMDP for the first scenario and discuss the modifications needed for the other two scenarios. The POMDP $\mathcal{M}_1 = (S, A, \delta, Obs, \Pi, \mu)$ is built as follows. The set of states $S = \mathcal{R} \times \mathcal{R} \cup \{\mathsf{Out}\}$ consists of pairs, where the components correspond to the current region of the quadrotor and the ground agent, respectively. The newly introduced state Out corresponds to the area outside of the grid. The set of actions are the actions available to the quadrotor, *i.e.,* $A = \{\mathsf{N}, \mathsf{E}, \mathsf{S}, \mathsf{W}, \mathsf{X}\}$. The transition relation combines the movement of the quadrotor and the motion model of the ground agent GA_u, *e.g.,* the probability $\delta((r_{i,j}, r_{x,y}), \mathsf{E})(r_{i',j'}, r_{x',y'})$ is the probability of the ground agent moving from region $r_{x,y}$ to region $r_{x',y'}$, while the quadrotor moves east from region $r_{i,j}$ is $r_{i',j'}$. Attempts of the quadrotor to move out of the grid lead with probability 1 to the newly introduced absorbing state Out. The set of observations $Obs = Obs_{\mathsf{Quad}} \cup \{o_{\mathsf{Out}}\}$ is given by the observations transmitted from the camera and an additional observation for state Out. The observation function Π is then directly given by the camera, *e.g.,* $\Pi(r_{1,1}, r_{2,1})(\mathsf{NW}) = 0.5$ and $\Pi(r_{1,1}, r_{2,1})(\mathsf{NE}) = 0.5$. For the state Out, we let $\Pi(\mathsf{Out})(o_{\mathsf{Out}}) = 1$. The initial state is sampled from the states, where the quadrotor's component is $r_{1,1}$.

The POMDP models $\mathcal{M}_2$ and $\mathcal{M}_3$ for the second and third scenario, respectively, are constructed analogously. Namely, $\mathcal{M}_2$ differs from $\mathcal{M}_1$ in the transition function that reflects the allowed movement of the ground agent, and $\mathcal{M}_3$ differs from $\mathcal{M}_1$ in the observation set and the observation function in order to correspond to the perfect camera $\mathsf{Cam}_{\mathsf{Perf}}$.

Objective. We consider a set of atomic propositions $AP = \{\mathsf{Detected}, A, B\}$. The labeling function L is defined over the set of states S of the POMDP as follows. For a state $(r_{i,j}, r_{x,y}) \in S$, we let $\mathsf{Detected} \in L((r_{i,j}, r_{x,y}))$ if and only if the quadrotor and the ground agent are in the same region of the grid, *i.e.,* $i = x$ and $j = y$. As expected, $A \in L((r_{i,j}, r_{x,y}))$ for $i = j = 1$, and analogously, $B \in L((r_{i,j}, r_{x,y}))$ for $i = j = 5$.

Recall that the objective for the quadrotor is to survey regions labeled with A and B, while avoiding the ground agent. The corresponding LTL formula over AP is

$$\square \lozenge A \ \wedge \ \square \lozenge B \ \wedge \ \square \neg \mathsf{Detected}.$$

The formula is translated to a deterministic parity automaton with 4 states and the synchronous product POMDP is constructed. Next, we present a crucial heuristic that makes qualitative analysis in all our examples feasible.

State space reduction. Intuitively, the algorithm in [4] that is used here to solve the parity POMDP problem, keeps track of the possible current regions of the ground agent, *i.e.,* keeps track of the current belief-support. For every possible belief-support, the algorithm implemented in [4] performs a computation that is exponential in the size of the belief-support. This becomes particularly time-consuming, when the ground agent is out of FOV of the camera. Consider for example the following case. Assume that the quadrotor hovers over region $r_{1,1}$ and the observation None is reported for 7 consecutive turns. It follows that the ground agent can be in any of the 21 regions outside of FOV, *i.e.,* the size of the belief-support is 21. To avoid the computational overhead, we implement the following heuristic. We modify the POMDP $\mathcal{M}_1$ so that we do not need to keep track of the ground agent whenever it is out of FOV. More formally, we construct a POMDP $\overline{\mathcal{M}_1}$, where for every region of the quadrotor $r_{i,j}$, we merge all the states $(r_{i,j}, r_{x,y})$ for which the ground agents' region $r_{x,y}$ is out of the FOV of the camera. Clearly, the ground agent in POMDP $\overline{\mathcal{M}_1}$ is more powerful than the ground agent in POMDP $\mathcal{M}_1$, as it can move arbitrarily among regions outside of the current FOV. However, the construction guarantees that any almost-sure winning policy in POMDP $\overline{\mathcal{M}_1}$ is also an almost-sure winning policy in POMDP $\mathcal{M}_1$.

4.2 Generated policies

To synthesize a finite-memory policy for the three considered scenarios, we used the implementation of [4] with the policy minimization technique designed in Sec. 3.2.

Scenario 1. In the scenario with an unrestricted ground agent GA_u and a quadrant camera $\mathsf{Cam}_{\mathsf{Quad}}$, there does not exist a finite-memory policy for the quadrotor that would satisfy the objective with probability 1. Intuitively, in non-corner regions of the grid, the quadrotor can always avoid the ground agent. However, it turns out that even though the ground agent moves probabilistically, it will eventually force the quadrotor into a corner and detect it. First, note that if the camera reports that the agent is in the FOV, there are at most two actions that the quadrotor can safely use. For example, if the quadrotor is in the center region $r_{3,3}$ and the camera reports observation NW because the agent is in region $r_{4,2}$, it is only safe to move south or east. If the ground agent from now on moves identically as the quadrotor, eventually the quadrotor enters corner region $r_{1,5}$. The

Grid dim.		$\lvert S\rvert,\lvert A\rvert,\lvert Obs\rvert$	Time	Policy, Size
5×5	Scenario 1	577/5/8	278.81s	✗/−
	Scenario 2	524/5/8	326.22s	✓/1287
	Scenario 3	409/5/12	20.12s	✓/703
6×6	Scenario 1	853/5/8	323.02s	✗/−
	Scenario 2	800/5/8	581.84	✓/2012
	Scenario 3	605/5/12	30.21s	✓/1052
7×7	Scenario 1	1185/5/8	610.94s	✗/−
	Scenario 2	1132/5/8	934.44s	✓/2887
	Scenario 3	524/5/12	46.86s	✓/1419
8×8	Scenario 1	1573/5/8	MO	−/−
	Scenario 2	1520/5/8	MO	−/−
	Scenario 3	1117/5/12	78.98s	✓/1964

Table 1: Results obtained for the three scenarios, and their versions on larger grids.

ground agent then moves to $r_{1,4}$ or $r_{2,5}$ with probability $1/2$, and the camera reports observation NW with probability $1/2$. In this case, the quadrotor must remain hovering in the corner and it is detected by the ground agent in the next move with positive probability. The discussed result is obtained for the POMDP $\overline{\mathcal{M}_1}$, but the argument for the non-existence of a finite-memory policy is valid also in POMDP $\mathcal{M}_1$.

Scenario 2. In the second scenario we combine a restricted ground agent GA_r with the quadrant camera $\mathsf{Cam}_{\mathsf{Quad}}$. Intuitively, we do not allow the ground agent to enter the corner regions to prevent the situation from the first scenario. Unlike in Scenario 1, there exists an almost-sure winning policy in this setting. The policy generated by the tool performs as follows. The quadrotor starts in A and tries to reach B, while avoiding the ground agent, and then uses the same approach to reach A starting in B and so on. First, it waits in A until it detects the agent in FOV and then chooses uniformly at random from all actions that are safe to use, *e.g.,* if the camera reports observations NE, the quadrotor remains hovering over A, but if it reports NW, the quadrotor moves with probability $1/2$ east and it remains in A with probability $1/2$. If it moves to a neighboring region and the agent is still in FOV, the next action is chosen in the same way as in the first step. On the other hand, if the agent moves out of FOV, the quadrotor can use the history of observations to either continue towards B or a different corner, *i.e.,* a safe location. However, it is only safe to make one step without the agent being in FOV of the camera, otherwise there is a positive probability of running into the agent. The policy minimization technique reduces the policy size by almost by a factor of two, e.g., in the case of the 5×5 grid, the state space reduces from 2033 to 1287 memory elements.

Scenario 3. In Scenario 3, we combine the unrestricted ground agent GA_u with the perfect camera $\mathsf{Cam}_{\mathsf{Perf}}$ that is able to track the position of the agent in FOV perfectly. In this case, there again exists an almost-sure winning policy. The synthesized policy proceeds in a similar way as the one for Scenario 2. In every step, if the agent is in FOV, the next move is chosen uniformly from all actions that are safe at the moment, *e.g.,* if the quadrotor is in region $r_{3,3}$ and the ground agent is in $r_{4,3}$, the quadrotor plays action N, E and S each with probability $1/3$ each. Unlike in Scenario 1, even if the quadrotor gets into a corner, the camera $\mathsf{Cam}_{\mathsf{Perf}}$ allows the quadrotor to escape. For example, if the quadrotor is in region $r_{1,5}$ and the ground agent is in $r_{2,4}$, the quadrotor will stay put until the agent moves either to region $r_{1,4}$ or $r_{2,5}$, and then the quadrotor will get out of the corner by moving

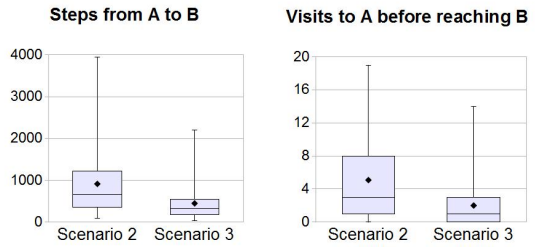

Figure 3: Evaluation of policies generated for Scenario 2 and 3. The statistics for each scenario is provided over 100 runs that start in region A and end with the first visit of region B.

to region $r_{2,5}$ or $r_{1,4}$, respectively. The policy minimization gain in this scenario is not so significant, for the 5×5 grid it reduces from 711 to 703 memory elements. The main reason is, that Scenario 3 contains much less uncertainty than Scenario 2, and the unminimized policy automaton is already state-efficient. The following remark summarizes the obtained results.

REMARK 1. *In Scenario 1, the quadrant camera* $\mathsf{Cam}_{\mathsf{Quad}}$ *is not informative enough to avoid the ground agent. Hence, there does not even exist a policy that wins with positive probability. In Scenario 2, we kept the quadrant camera and tried to restrict the ground agent such that it cannot enter the corners of the grid. This turned out to be sufficient and there exists an almost-sure winning policy. In the last scenario, we replaced the camera with a more informative version* $\mathsf{Cam}_{\mathsf{Perf}}$ *and the results show that there exists an almost-sure winning policy even in the case, where the ground agent is not restricted and can enter the corner regions.*

In Tab. 1, we provide running times and sizes of obtained policies for all three scenarios. To demonstrate the scalability of the game solving tool, we considered the three scenarios on larger grids. In Tab. 1, we list running times for grids $5 \times 5, 6 \times 6$, 7×7, and 8×8. All computations were run on a quad-core i7 processor with 8 GB of RAM.

4.3 Simulation

We simulated the obtained policies for Scenario 2 and 3 in Matlab. For each of the two cases, we executed 100 runs that start in region A and end with the first visit of region B. In Fig. 3, we evaluate the number of steps that were needed to reach B, *i.e.,* the length of the resulting runs, and the number of returns to A before B was reached. In Scenario 3, where the camera provides more information about the agent's location, region B was reached faster and with lower number of returns to A.

Note that even though the quadrotor is guaranteed to satisfy the mission in Scenario 2 and 3 under the generated policies, the number of steps needed to move from one region of interest to another, *i.e.,* from A to B and vice versa, is high. The reason is that in both policies, the next action for the quadrotor is always chosen randomly from all actions that are safe under the current observation, even though some of the actions obviously lead to progress while others move the quadrotor further away from its next goal region. In our future work, we aim to look at the problem of generating policies for POMDPs that not only provably guarantee the satisfaction of given temporal constraint but also optimize progress towards the satisfaction, *e.g.,* in our case the number of steps needed to move from A to B.

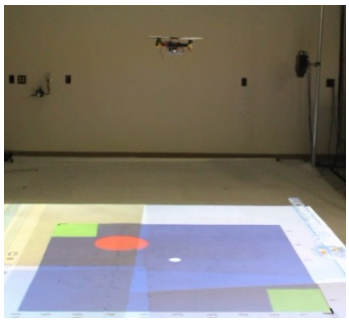

Figure 4: Top: Robotic testbed used to demonstrate runs of the quadrotor in all three scenarios. The ground agent is simulated with a red circle. For better visualization, the quadrotor's current position on the grid is marked with a small white circle.

4.4 Experiments on robotic platform

Besides simulation, we also used a robotic testbed to demonstrate runs of the quadrotor in the environment. The robotic testbed consists of a quadrotor flying above a square area on which the grid of 5 by 5 regions is projected using four projectors (see Fig. 4). The ground agent is simulated as a large red circle that is being projected on the ground together with the grid. The quadrotor is equipped with a camera that uses an image processing procedure according to the chosen scenario. In Fig. 5, we depict a sample image taken by the camera during execution and the resulting image after color recognition. For better visualization, we also depict the division of the image into sectors. For the quadrant camera, the resulting observation is given as uniform distribution over quadrants that contain white color, or the central sector if it contains the majority of white. For the perfect camera, the observation is given by the sector that contains the majority of white. In both cases, the camera reports the agent to be out of FOV if the image after color recognition does not contain any white. The videos from the demonstration are available at

`http://www.fi.muni.cz/~x175388/demonstrationHSCC15.html`.

For Scenario 1, we demonstrate a run, where the agent forces the quadrotor into a corner and detects it. For Scenario 2 and 3, we demonstrate a run using the winning policies.

5. CONCLUSION AND FUTURE WORK

In this work, we used a theoretical tool to control a quadrotor aiming to satisfy a temporal mission in a probabilistic and partially observed, dynamic environment. The obtained control policies are provably correct with respect to the considered mission, but they do not progress towards the mission satisfaction fast enough. In our future work, we aim to use the insight gained from the case study to look at the problem of generating policies that not only satisfy the mission but also provide a guarantee on the rate of progress.

Acknowledgment. We thank Austin Jones (Boston University, MA, USA, `austinmj@bu.edu`) for his help with image processing for the camera models and for helpful discussions.

6. REFERENCES

[1] C. Baier, M. Größer, and N. Bertrand. Probabilistic omega-automata. *J. ACM*, 59(1), 2012.

[2] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

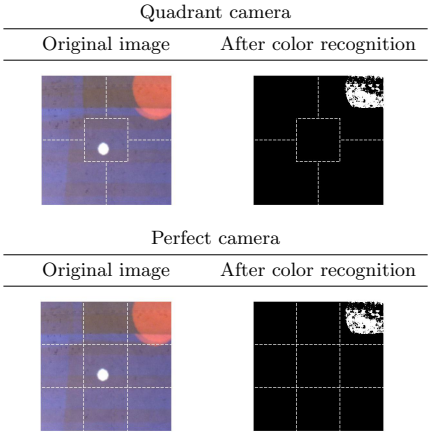

Figure 5: A sample image taken by the camera during execution and the result after color recognition.

[3] A. Brooks, A. Makarenko, S. Williams, and H. Durrant-Whyte. Parametric POMDPs for planning in continuous state spaces. *J. RAS*, 54(11), 2006.

[4] K. Chatterjee, M. Chmelik, R. Gupta, and A. Kanodia. Qualitative Analysis of POMDPs with Temporal Logic Specifications for Robotics Applications. *CoRR*, abs/1409.3360, 2014.

[5] K. Chatterjee, M. Chmelik, and M. Tracol. What is Decidable about Partially Observable Markov Decision Processes with ω-Regular Objectives. In *CSL*, 2013.

[6] E. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT press, 1999.

[7] K. Hsiao, L. Kaelbling, and T. Lozano-Perez. Grasping POMDPs. In *ICRA*. IEEE, 2007.

[8] D. Hsu, W. S. Lee, and N. Rong. A point-based POMDP planner for target tracking. In *ICRA*, 2008.

[9] T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM J. Comput.*, 1993.

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

[11] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 1987.

[12] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, 2003.

[13] N. Piterman. From nondeterministic Buchi and Streett automata to deterministic parity automata. In *LICS*, 2006.

[14] N. Roy and S. Thrun. Coastal Navigation with Mobile Robots. In *NIPS*, 1999.

[15] S. Safra. On the complexity of ω-automata. In *FOCS*, 1988.

[16] G. Shani, P. Poupart, R. I. Brafman, and S. E. Shimony. Efficient ADD Operations for Point-Based Algorithms. In *ICAPS*, 2008.

[17] T. Smith and R. Simmons. Point-Based POMDP Algorithms: Improved Analysis and Implementation. In *UAI*, 2005.

[18] M. Spaan and N. Vlassis. A point-based POMDP algorithm for robot planning. In *ICRA*. IEEE, 2004.

[19] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, 2005.

[20] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state systems. In *FOCS*. IEEE, 1985.