A Parametric Model for Near-Optimal Online Synthesis with Robust Reach-Avoid Guarantees

Mario Gleirscher

Philip Hönnecke

April 2, 2025

Abstract

Objective: To obtain explainable guarantees in the online synthesis of optimal controllers for high-integrity cyber-physical systems, we reinvestigate the use of exhaustive search as an alternative to reinforcement learning. *Approach:* We model an application scenario as a hybrid game automaton, enabling the synthesis of robustly correct and near-optimal controllers online without prior training. For modal synthesis, we employ discretised games solved via scope-adaptive and step-pre-shielded discrete dynamic programming. *Evaluation:* In a simulation-based experiment, we apply our approach to an autonomous aerial vehicle scenario. *Contribution:* We propose a parametric system model and a parametric online synthesis.

1 Introduction

Motivation. To achieve complex and critical tasks, systems with a high grade of autonomy perform decision making and control at strategic and tactical levels under sparse human-machine interaction. Consider, for example, navigation of autonomous aerial vehicles (AAVs), as illustrated in Figure 1. At the tactical level (e.g., route segment tracking), often control problems of non-linear, disturbed dynamical systems have to be solved by constructing provably robust (i.e., for safety) and near-optimal (i.e., for minimum-cost reachability) controllers. To accommodate uncertainties (e.g., changing environments) and perform at scale, this type of controller synthesis is preferably done online, that is, during operation and right before the actual use of the controllers.

Challenges. Reinforcement learning (RL), a common and versatile type of approximate dynamic programming (ADP) frequently used in online synthesis, has shown to have several assurance-related drawbacks:

• partial, imprecise, or even missing guarantees (due to a lack of behavioural coverage or an inappropriate initialisation [4, Sec. 6]),



Figure 1: Scene model for an AAV following a route \bar{r} from waypoint \mathbf{p}_0 to \mathbf{p}_7

- post-hoc explanations (i.e., identifying failure causes may require training even with transparent algorithms [10]),
- lack of robustness to changing operational profiles (e.g., obstacle scenarios not covered or differing from training setups [18, p. 30]), and
- limited or costly training (off-line/simulation or on-line/real interaction; requires active supervision, e.g., real-time safety shielding [20, 13]).

These limitations together with a need for more precise and complete guarantees revive exhaustive, hence, less efficient forms of synthesis. However, to keep computation within bounds while guaranteeing critical properties, online synthesis, whether or not based on RL, implies restrictive trade-offs (e.g., predictive imprecision, latencies in segment tracking).

Research Question. Not focusing on real-time performance, what kind of robust reach-avoid guarantees can exhaustive numerical algorithms for near-optimal online synthesis provide and under which assumptions?

Approach and Application. We focus on an autonomous aerial vehicle (AAV)based delivery scenario (Figure 2) formalised as a hybrid game automaton (HGA). The HGA enables tactical controller synthesis to be done online (i.e., during operation) via discrete dynamic programming (DDP). We permit a perforated¹ fixed obstacle cloud and bounded uncertain wind disturbance. Unsafe actions possibly enabled by the discretisation are filtered during DDP before finishing a search stage, which corresponds to pre-shielding. A *supervisor* [5], a high-level controller, coordinates the interruption of the tactical controller by an obstacle evasion unit if moving obstacles are about to intrude the planned trajectory.

Related Work. Online synthesis of controllers for autonomous systems has been a research subject for many decades, with a steady activity around robustly correct, near-optimal control. Below, we highlight some more recent works.

Real-time-capable online synthesis techniques have been proposed for specialpurpose control of single- and multi-agent systems. For example, Li et al. [15]

 $^{^1\}mathrm{Start}$ and end of a given route must be connected with a wide enough tube.



Figure 2: Assured online synthesis with (a) remote or (b) on-board computation

apply a fast model-predictive control scheme focusing on stabilisation around a given continuous reference trajectory. They use a quadratic cost term enabling initialised, sequential quadratic programming. For AAV collision avoidance, Bertram et al. [2] and Taye et al. [19] realise multi-agent reachability-based state space reduction to accelerate backward Markov decision process (MDP) policy search. In comparison, our approach is significantly slower but more widely applicable, allowing non-convex cost terms, sparse reference trajectories, near-optimality and guaranteed reach-avoidance under bounded disturbance, and it is integrated into a flexible multi-tier hybrid control scheme.

Scalable offline synthesis techniques have been developed for discrete and hybrid single- and multi-agent systems. For example, Ivanov et al. [12] apply deep-RL to compute contract-compliant controllers for each mode of a hybrid automaton. Gu et al. [8] use RL to perform synthesis at scale for timed stochastic multi-agent systems. In [17], we refine a stochastic Petri-net abstraction with partial state observability to synthesise optimal schedules for tasked robot collectives. By approximating fixpoints and sacrificing stochasticity in the weighted reach-avoid game setting, we bypass an even higher complexity of exact MDP synthesis. This enables us to pre-process up to 200 Mio. states in non-real-time, going far beyond the state space in [17] and yet in a time scale similar to approaches, such as [8, 12]. Our work provides reach-avoid guarantees without a potentially intense training needed for RL. However, because of the scenario-sensitive parameterisation, our technique can currently not be used in fast real-time settings, such as autonomous driving or for collision avoidance [15, 12].

Contributions. Our approach enhances previous work as follows:

(i) Step-shielded online synthesis: We provide an algorithm for solving discretised modal games with reach-avoid winning conditions. The algorithm uses step-shielded DDP and supports non- and quasi-stationary policies (increasing robustness to disturbances at the cost of performance) [14]. We slightly reduce DDP's curse-of-dimensionality problem by scope adaptation² and fixpoint approximation for the winning region.

 $^{^{2}\}mathrm{The}$ relevant fraction of the state space is selected according to the current system state

(ii) Parametric hybrid game model: We construct a parametric weighted HGA covering a range of typical AAV scenarios. The automaton enables a three-tier separation to allow a trading off of operational cost and flexibility: higher-level supervision (e.g., moving obstacle evasion), near-optimal trajectory tracking, and sub-tactical control enabling local policy learning (e.g., via deep-RL). We embed our algorithm into a player for the HGA.

A preliminary variant of (i) and (ii) was implemented and evaluated in [11].

Outline. After the preliminaries (Section 2), we describe our AAV application and control problem (Section 3). Our contributions to online synthesis with guarantees follow in the Section 4. We evaluate our approach experimentally (Section 5) and close with a discussion and remarks (Sections 6 and 7).

2 Preliminaries

Notation. For a set of variables $Var = X \uplus U \uplus D$, let $\mathbb{X} \subset \mathbb{Z}^X$ be an X-typed, finite, *m*-dimensional Euclidean state space,³ and let \mathcal{U} and \mathcal{D} be U- and D-typed control and disturbance ranges, each including **0**. Moreover, let the classes of terms $\mathcal{T}(Var)$ and constraints $\mathcal{C}(Var)$ over Var. $\|\cdot\|$ is the corresponding 2-norm, $A \oplus B = \{a + b \mid a \in A \land b \in B\}$ the Minkowski sum of $A, B \subseteq \mathbb{X}$, and $A|_{Var'}$ the projection of tuples in A to variables in $Var' \subseteq Var$,⁴ and $-\rightarrow$ indicates a partial map. Point-wise operators are lifted as usual.⁵ Moreover, \underline{I} and \overline{I} denote the lower and upper bounds of an interval $I \subseteq \mathbb{R}$ and $\underline{I}...\overline{I}$ signifies that $I \subseteq \mathbb{Z}$. $\llbracket \varphi \rrbracket_{\mathfrak{M}}$ denotes φ 's models in domain \mathfrak{M} ,⁶ formally, $\llbracket \varphi \rrbracket_{\mathfrak{M}} = \{M \in \mathfrak{M} \mid M \models \varphi\}$. Given the finite sequences \mathbb{X}^* over \mathbb{X} , the length $|\overline{x}|$ of a sequence $\overline{x} \in \mathbb{X}^*$, and its value $\overline{x}_k \in \mathbb{X}$ at position $k \in 1...|\overline{x}|$, we use $\overline{\mathbb{X}}(\Delta), \mathbb{X} \subset \mathbb{X}^*$ to denote the classes of trajectories with any two subsequent states inside some $\Delta \subseteq \mathbb{Z}^m$ and with $\mathbb{X} = \overline{\mathbb{X}}([\pm 1]^m)$.

Weighted Hybrid Game Automata. Given a set of modes Q, an alphabet A, events $E \subseteq Q \times A \times Q$, and the hybrid state space $S = Q \times X$, a weighted HGA G = (Gra, Var, Ini, Inv, f, Jmp, F) comprises a mode transition graph Gra = (Q, A, E), initial conditions $Ini: Q \to C(X)$, invariants $Inv: Q \to C(X)$, flow constraints $f: S \to C(Var \cup \dot{X})$, jump conditions $Jmp: E \to C(Var \cup X^+)$ comprised of guards and updates, and weighted winning conditions $F: S \to 2^{\mathcal{T}(Var)}$, generalising final conditions $Fin: Q \to C(X)$ [9]. The copies \dot{X} and X^+ refer to the time derivatives and discrete updates of X. grd(e) and upd(e) denote the guard and update conditions of Jmp(e). In the remainder, we frequently use $\mathbf{s} = (q, \mathbf{x}) \in S$ for a state of G.

and extended on-the-fly to increase solvability of the online synthesis problem.

³For a 3D coordinate **p**, we use the typical naming convention $\mathbf{p} = (x, y, z)^{\intercal}$.

 $^{{}^{4}\}mathrm{We}$ omit set parentheses when referring to singleton sets in subscripts.

 $^{{}^{5}}$ For example, min, max on vectors of sets are evaluated element-wise and return a vector of scalars. Negation of a set returns the set of negated elements.

⁶For example, the class of states, state pairs, sequences, or transition systems.

Integer Difference Games. Let $\Pi_{\mathcal{Y}} = \mathbb{X} \times \mathbb{N}_+ \dashrightarrow \mathcal{Y}$ be a strategy space for a player with alphabet \mathcal{Y} . Given a flow constraint f and its discretised Taylor expansion \hat{f} , we associate a state \mathbf{s} of G with a discretised two-player game $G_{\mathbf{s}} = (\mathcal{X}, \hat{f}, F)$.⁷ We use $\mathbf{u} : \Pi_{\mathcal{U}}$ and $\mathbf{d} : \Pi_{\mathcal{D}}$ to describe the memory-less strategy profile of the two players, the controller and the environment, based on the non-empty control and disturbance ranges \mathcal{U} and \mathcal{D} . Then, for a horizon Nand the winning condition $F = (L, \Phi)$ with stage and terminal costs L and Φ , a finite-horizon, discrete optimal controller $\mathbf{u}^* : \Pi_{\mathcal{U}}$ can be obtained from solving a constrained, discrete dynamic optimisation problem [3]. Such a problem can be solved with a forward-Euler DDP (Algorithm 1) for the period I = 1..N.

Algorithm 1 Discrete dynamic programming								
1: procedure dDP(in $G_{\mathbf{s}}, I$; out V, \mathbf{u}^*)								
2:	$ (V, \mathbf{u}^*) \leftarrow \perp^{ \mathcal{X} \times I} $	$\perp \dots$ undefined or maximal, e.g., \top						
3:	$V(\mathcal{X},\bar{I}) \leftarrow \Phi(\mathcal{X},\bar{I})$	\triangleright initialise terminal cost						
4:	for $k \in [\underline{I}, \overline{I} - 1]$ do	\triangleright implies backward iteration						
5:	$\mathbf{for} \ \mathbf{x} \in \mathcal{X} \ \mathbf{do}$							
6:	$ \mathbf{x}^{\mathcal{UD}} \leftarrow \mathbf{x} + \hat{f}(\mathbf{x}, \mathcal{U}, \mathcal{D}, k) $	\triangleright forward unit Euler set						
7:	$ V(\mathbf{x},k) \leftarrow \min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \left\{ L(\mathbf{x},u,d,k) + V(\mathbf{x}) \right\} $	$^{ud}, k+1)\} riangle value$						
8:	$\mathbf{u}^{*}(\mathbf{x},k) \leftarrow \arg\min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \left\{ L(\mathbf{x},u,d,k) + \right\}$	$V(\mathbf{x}^{ud}, k+1)\}$						
9:	$\begin{bmatrix} \\ \\ \end{bmatrix}$ if $\widehat{fp}_{dDP}(k)$ then break	\triangleright approximate fixpoint found?						

The single-step successor $\mathbf{x}^{ud} = \mathbf{x} + \hat{f}(\mathbf{x}, u, d, k)$ is lifted in Line 6, such that $\mathbf{x}^{ud} \in \mathbf{x}^{\mathcal{UD}} \subseteq \mathbb{X}$. Moreover, let $\mathbf{x}^{ud} \in \bar{\mathbf{X}}$ be the *N*-step successor (trajectory) of *G* emanating from $\mathbf{x} \in \mathbb{X}$ under influence of (\mathbf{u}, \mathbf{d}) , and $\mathbf{x}^{\mathbf{uD}} \subseteq \bar{\mathbf{X}}$ be the family of such trajectories under \mathcal{D} . $V(\mathbf{x}, k) \in [0, \top]$ with $\top < \infty$ is the finite-horizon value corresponding to the optimal cost-to-go, using the optimal profile $(\mathbf{u}^*, \mathbf{d}^*)$. Line 9 allows a check \hat{fp}_{dDP} for whether a fixpoint is approximated prior to reaching *N*. While the actual fixpoint yields a stationary optimal strategy profile, a fixpoint approximation still guarantees bounded correct (quasi-stationary and near-optimal) strategies under reduced memory consumption and computational effort. Let \mathcal{G} be the class of integer difference games.

3 Aerial Delivery as a Hybrid Game

Assumptions. Let $X = {\mathbf{p}, \mathbf{v}, i}$ with the AAV position⁸ $\mathbf{p} : \mathbb{Z}^3$ and velocity $\mathbf{v} : [\pm v_{\max}]^3$ (with an absolute maximum at v_{\max}) and a next-waypoint index $i : \mathbb{N}$. We use convex sets $\mathcal{X}_{\mathbf{s}} \subseteq \mathbb{X}$, called scopes,⁹ and $\mathcal{P} = \mathbb{X}|_{\mathbf{p}}$, the position grid of the scenery. Given \mathbb{X} , an AAV task $T = (\mathbb{X}, \bar{r}, \mathcal{O})$ comprises a fixed obstacle cloud $\mathcal{O} \subset \mathcal{P}$ and a route $\bar{r} = {\mathbf{p}}_i \}_{i=1}^n$ where the predicate valid(\bar{r}) requires \bar{r} to

⁷Players share the state and time but do not know each others' future actions.

 $^{{}^{8}\}mathbf{p}|_{z}$ measures distance above local ground outside buildings.

 $^{{}^{9}\}mathcal{X}_{\mathbf{s}}$ is not included as a state variable in X because it can be derived from $\mathbf{x}.$



Figure 3: Overall AAV control block structure



a: Parametric tactical control logic $G[\mathbf{U}, T]$ b: Supervisor logic

Figure 4: Tactical AAV control (f/Inv are above/below the dashed lines in the modes, and Jmp is shown in the transition labels) and supervisor logic

have $n \geq 4$ different waypoints $\mathbf{p}_i \in \mathcal{P}$ with $\mathbf{p}_{1,n}$ being on the ground, $\mathbf{p}_{2,n-1}(x, y)$ -superimposed, and $\mathbf{p}_{2..n-1}$ residing above a minimum height z_{\min} . Given a cube $\Delta_{\delta} = [\pm \delta]^3$ for a small $\delta \in \mathbb{N}_0$, we require \mathcal{O} to be δ -perforated, such that

$$\exists \bar{x} \in \tilde{\mathbb{X}} \colon \bar{r} \subseteq \underbrace{\bar{x}}_{\text{cont.}} \underbrace{\bar{\lambda}}_{\text{b}} \oplus \Delta_{\delta} \wedge \bar{x}|_{\mathbf{p}} \oplus \Delta_{\delta} \cap \mathcal{O} = \varnothing \,. \tag{1}$$

Informally, the route is enclosed in a tube with radius δ not colliding with fixed obstacles. We assume (\bar{r}, \mathcal{O}) to be delivered by map- and SLAM-based navigation and sensor fusion on-board the AAV (Figure 3) and allow (\bar{r}, \mathcal{O}) to be updated at every \mathbf{p}_i invariant under (1) and valid (\bar{r}) .¹⁰

Tactical Control. We consider a HGA G (Section 2) combining the logic of the AAV route and segment tracking units, where Gra, Inv, f, and Jmp are illustrated in Figure 4a and Ini(q) = T for $q \in Q$. For brevity, (1) and $valid(\bar{r})$ as global invariants remain implicit in Ini and Inv. To compute the current

¹⁰In Figure 4 and below, we use predicates over $\mathcal{O}, \mathcal{O}_m, \bar{r}$, and **u**. \mathcal{O} and \bar{r} are game parameters, **u** is a parameter of the tactical control's alphabet (Section 4), and \mathcal{O}_m and **u** are parameters of the supervisor's alphabet.

scope, we use the function

$$\mathcal{X}_{\mathbf{s}} = \begin{cases} (\mathbf{p}_{i}, \mathbf{0}) \oplus ([\pm \delta_{p,q}]^{2} \times [0, \max\{\mathbf{p}, \mathbf{p}_{i}\} \quad q \in \{\text{depart}, \\ +\delta_{p,q}] \times [\pm \delta_{v,q}]^{3}) \times \{i\}, & \text{arrive, standby} \} \\ ([\min\{\mathbf{p}, \mathbf{p}_{i}\}, \max\{\mathbf{p}, \mathbf{p}_{i}\}] \times \{\mathbf{0}\}) \oplus & q \in \{\text{cruise}\} \\ ([\pm \delta_{p,q}]^{3} \times [\pm \delta_{v,q}]^{3}) \times \{i\} \end{cases}$$
(2)

describing *transition cuboids* (Figure 1 left and right) around route segments for moments when the AAV is near one waypoint and proceeds to the next. We use $\delta_{p,q}$ and $\delta_{v,q}$ for position and velocity padding of mode q and the abbreviations $\operatorname{in}(\mathcal{X}) \equiv \mathbf{x} \in \mathcal{X}, \ \mathbb{X}_0 \equiv \{\mathbf{x} \in \mathbb{X} \mid \mathbf{p}|_z = 0 \land \mathbf{v} = \mathbf{0} \land i = 1\}, \ \mathbf{s}_q^{++} = (q, (\mathbf{p}_i, \mathbf{0}, i + i)) \land \mathbf{v}_q^{-1} \in \mathcal{X}\}$ $(1)^{\mathsf{T}}$, i^{11} and $i^{++} \equiv i^{+} = i + 1$.

The AAV dynamics¹² in all modes, except for standby, is given by

$$fly = \dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\mathbf{v}} \\ \dot{i} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{u} + \mathbf{d} + \mathbf{g} \\ 0 \end{bmatrix}$$
(3)

with an implicit unit mass (i.e., vehicle + payload = 1), an approximate¹³ gravitational force $\mathbf{g} = (0, 0, -10)^{\intercal}$, and $\mathbf{u} \in \mathcal{U}, \mathbf{d} \in \mathcal{D}$ for $\mathcal{U}, \mathcal{D}: 2^{\mathbb{Z}^3}$. In standby mode, the flow condition is $stable \equiv \dot{\mathbf{x}} = \mathbf{0}$.

In our experiments, we use small $\mathcal{U}, \mathcal{D} \subseteq [\pm 2]^3$ to allow the AAV to accelerate in 26 directions and wind disturbance to occur in 4 directions, as well as $v_{\rm max} =$ 10 ranging down to 5 for efficiently handling smaller scopes.

Supervisory Control. Strategic AAV control comprises a supervisor (Figure 4b), which separates route and segment tracking from moving-obstacle evasion. The supervisor interrupts tactical control and resumes it after an evasion manoeuvre. Encoded in F, for all modes but standby, the unsafe set encompasses (inevitable) collisions with fixed obstacles as¹⁴

$$\operatorname{coll}_{\mathsf{f}}(\mathcal{O}) \equiv \exists \mathbf{o} \in \mathcal{O} \colon \|\mathbf{o} - \mathbf{p}\| \le \delta, \quad \text{if } q \neq \mathsf{standby}.$$

$$\tag{4}$$

For a set $\tilde{\mathcal{O}}_m$ of trajectories of moving obstacles tracked by the supervisor,

$$\operatorname{coll}_{\mathsf{m}}(\mathbf{u}, \tilde{\mathcal{O}}_{\mathsf{m}}) \equiv \exists k \in I, \bar{o} \in \tilde{\mathcal{O}}_{\mathsf{m}} \colon \mathbf{x}_{k}^{\mathbf{u}\mathcal{D}}|_{\mathbf{p}} \cap \bar{o}_{k} \oplus \Delta_{\delta} \cap \mathbf{p} \oplus \Delta_{sbd}(\mathbf{v}) \neq \emptyset$$

indicates that the predicted trajectory \bar{o} of some moving obstacle crosses the AAV trajectory $\mathbf{x}^{\mathbf{u}\mathcal{D}}|_{\mathbf{p}}$ within safe braking distance Δ_{sbd} . Evasion manoeuvres (i.e., replacing **u** by an evasive \mathbf{u}_e) could be computed by Algorithm 1. However, there are other ways of computing $\mathbf{x}^{\mathbf{u}\mathcal{D}}|_{\mathbf{p}}$ efficiently (e.g., [1, 20]) and \mathbf{u}_e with

¹¹Note how \mathbf{s}_q^{++} refers to the next waypoint \mathbf{p}_i and the one after that, \mathbf{p}_{i+1} . \mathbf{s}_q^{++} will be used to compute the goal region for a modal game.

¹²At tactical control level, we employ a simplification of the dynamics to a point mass.

 $^{^{13}\}mathbf{g}$ is omitted in the isolated dynamics. The compensation of numerical imprecision can be delegated to lower-level stability control based on system identification. ¹⁴With $\|\mathbf{o} - \mathbf{p}\| \leq \delta$ instead of $\mathbf{p} \in \mathbf{o} \oplus \Delta_{\delta}$, we can work with $\delta = 1.5$ in experiments.

guarantees (e.g., [16]). To simplify our setting, we assume $\tilde{\mathcal{O}}_{m} = \emptyset$ and refer to the broader treatment of supervision in, for example, [5].

Let us now formulate the synthesis problem focused on in this work.

Definition 1 (Online Synthesis Problem) Given a hybrid game automaton G and a task T, continuously find tactical controllers **u** steering the system along route \bar{r} while safely circumventing unsafe regions. For correctness, we need **u** to be (i) δ -robust (i.e., safe under bounded disturbance D), (ii) nearoptimal (i.e., follow the minimum-cost path inside padded segment scopes, approximating all waypoints), and (iii) reaching the endpoint of \bar{r} .

4 Online Synthesis via Parametric Games

Parametric Modal Games. We assume that Gra is controllable,¹⁵ such that G reduces to a parametric reach-avoid integer difference game $G_{\mathbf{s}} = (\mathcal{X}_{\mathbf{s}}, \hat{f}, F)$ to be solved for and played after a jump to \mathbf{s} . The forward-Euler scheme in Line 6 of Algorithm 1 uses an isolated¹⁶ variant of the right-hand side of \hat{f} . For $F = (L, \Phi)$, we employ

$$L(\mathbf{u}, \mathbf{d}; \mathbf{s}, k) = \begin{cases} 0, \text{ if } \rho \wedge \neg \alpha \\ \top, \text{ if } \alpha \text{ (shielded)} \\ \lambda(\mathbf{u}, \mathbf{d}; \mathbf{x}, k), \text{ otherwise} \end{cases} \quad \Phi(\mathbf{s}) = \begin{cases} 0, \text{ if } \rho \wedge \neg \alpha \\ \top, \text{ otherwise} \end{cases} \quad (5)$$

where ρ and α specify the goal and unsafe regions to be reached and avoided, respectively. We assume that $\llbracket \rho \rrbracket \subseteq \mathcal{X}_{\mathbf{s}}$ and $\llbracket \alpha \rrbracket \subseteq \mathbb{X}$ with $\llbracket \rho \rrbracket \setminus \llbracket \alpha \rrbracket \neq \emptyset$. Furthermore, $\lambda(\mathbf{u}, \mathbf{d}; \mathbf{x}, k) = \mathbf{x}^{\mathsf{T}} P \mathbf{x} + \mathbf{u}^{\mathsf{T}} Q \mathbf{u} + \mathbf{d}^{\mathsf{T}} R \mathbf{d}$ is a weight term with correspondingly dimensioned matrices P, Q, and R.

L penalises α and rewards ρ , maximally. Given (1), (5) implies $V(\mathbf{0}, \cdot) = 0$. Not shown here, *k* can be used in λ to penalise run-time. Overall, the call $dDP(G_{\mathbf{s}}, I)$ (Algorithm 1) provides a near-optimal controller $\mathbf{u}_{\mathbf{s}}^{*}$.¹⁷

Bounded Correctness via Approximate Value Fixpoints. Let $W(\mathcal{X}_{\mathbf{s}}, k) = \{\mathbf{x} \in \mathcal{X}_{\mathbf{s}} \mid V(\mathbf{x}, k) < \top\}$ be an approximation of $G_{\mathbf{s}}$'s winning region at time k when playing no more than N - k steps. In particular, $W(\mathcal{X}_{\mathbf{s}}, N) = \{\mathbf{x} \in \mathcal{X}_{\mathbf{s}} \mid \Phi(\mathbf{x}) < \top\}$. To reduce the computational effort in Algorithm 1, we employ a condition $\widehat{fp}_{\text{dDP}}$ for premature termination in Line 9. It is defined as

$$\widehat{fp}_{\rm dDP}(k) \equiv |W(\mathcal{X}_{\mathbf{s}}, k)| = |W(\mathcal{X}_{\mathbf{s}}, k+1)| \lor \widehat{fp}_{\mathbf{U}}(k), \qquad (6)$$

where the conjunct

$$\widehat{fp}_{\mathbf{U}}(k) \equiv \exists k' \ge k \colon \top \notin V(\mathbf{x} \oplus \Delta_{\delta}, k')$$

 $^{^{15}\}mathrm{All}$ jumps are solely controlled or chosen by the system.

¹⁶E.g., (3) without **g** and under a change of variables via $\mathbf{x}_{iso} = \mathbf{x}_{orig} - (\mathbf{p}_i, \mathbf{0}, 0)^{\intercal}$.

¹⁷Deviating from Section 2, we pass q to dDP by calling L and Φ with $\mathbf{s} = (q, \mathbf{x})$.

ensures that, latest at step k, **x** is δ -robustly located inside $W(\mathcal{X}_{\mathbf{s}}, k)$. This approximation allows us to check whether the number of states with a *robustly safe* and goal-reaching trajectory stabilises at k and, hence, within the horizon N. Fixpoint approximation safes time but it limits our approach to bounded correctness and increases the deviation of the discrete solutions from optimality.

Scope Adaptation. To further reduce computational effort, we use Algorithm 1 with a spatio-temporal *scope extension*. The result is Algorithm 2, with the aim to increase solvability of $G_{\mathbf{s}}$ by checking $\hat{f}_{p_{\mathbf{U}}}(1)$ in Line 7.

Algorithm 2 DDP with scope extension (in Line 4, note that $\mathbf{s} = (q, \mathbf{x})$)

1: procedure $\mathbf{U}_{\delta}(\text{in } G_{\mathbf{s}}, I; \text{ out } V^{\overline{I}'}, \mathbf{u}^*, \mathcal{X}', I')$ 2: $(\mathcal{X}', I', \Delta'_{\mathcal{X}}, \delta_{I}') \leftarrow (\mathcal{X}_{\mathbf{s}}, I, 0, 0) \quad \triangleright \text{ initialise scope and prediction interval}$ 3: repeat 4: $|(\mathcal{X}', I') \leftarrow (\mathcal{X}' \oplus \Delta'_{\mathcal{X}}, \underline{I}'..\overline{I}' + \delta_{I}') \quad \triangleright \text{ extend scope (not initially)}$ 5: $|(V^{\overline{I}'}, \mathbf{u}^*) \leftarrow \text{dDP}((\mathcal{X}', \hat{f}, F), I') \quad \triangleright \text{ compute controller, Algorithm 1}$ 6: $|(\Delta'_{\mathcal{X}}, \delta_{I}') \leftarrow (\Delta_{\mathcal{X}}, \delta_{I}) \quad \triangleright \text{ use hyper-parameters } \Delta_{\mathcal{X}}, \delta_{I}$ 7: $|\text{untill } \hat{fp}_{\mathbf{U}}(1) \quad \triangleright \text{ is x robustly located in the 1-winning region?}$ 8: $_\text{return } (V^{\overline{I}'}, \mathbf{u}^*, \mathcal{X}', I')$

Hyper-Policies and Pre-Computation. Algorithm 2 realises a hyper-policy $\mathbf{U}_{\delta} : \mathcal{G} \times 2^{\mathbb{N}_+} \to (\mathbb{X} \times \mathbb{N}_+ \dashrightarrow \mathcal{U})$ for the parametric $G[\mathbf{U}, T]$ in Figure 4a. $\mathbf{u}_{\mathbf{s}}^* = \mathbf{U}_{\delta}(G_{\mathbf{s}}, I)$ is defined for $\mathcal{X}' \times I' \subset \mathbb{X} \times \mathbb{N}_+$ and solves $G_{\mathbf{s}}$ by determinising the $(\mathcal{U}, \mathcal{D})$ -underspecified modal dynamics of G. $\mathbf{u}_{\mathbf{s}}^*$ is intended to be computed online, remotely or on-board (Figure 2), if $\mathbf{x} \in Inv(q)$. Any suffix of \bar{r} after \mathbf{p}_i can be updated together with the corresponding update of $\mathbf{u}_{\mathbf{s}}^*$.

A Hybrid Game Player. Algorithm 3 combines Algorithm 2 with an execution routine for HGAs. A play of $G_{\mathbf{s}}$ is a co-execution of both players successively applying a strategy profile, presumably $(\mathbf{u}^*, \mathbf{d})$, to the dynamics \hat{f} . The loop (Lines 3 to 10) plays $G_{\mathbf{s}}$ starting from $\mathbf{x}_0 = \mathbf{s}|_{\mathbf{x}}$. For the possible jump in Line 5, the \exists of the **switch** is to be read as "check and pick $e^{n.18}$ The numerical integrator in Line 9 describes the simultaneous inputs $(\mathbf{u}^*, \mathbf{d})$ of both players being used in the dynamics and adds the corresponding increment to the current state \mathbf{x} , resulting in a hybrid trajectory $\mathbf{\bar{s}} \in \bar{S}$.

Execution (Line 10) is constrained by several conditions: The first, Ω , specifies termination. The other three, $\mathbf{x} \notin W(\mathcal{X}_{\mathbf{s}}, k), k \geq N$, and $\neg Inv(q)$, specify failure (i.e., unsafe behaviour, $\mathbf{x} \notin [\rho \land \neg \alpha]$ if k = N), timeout (beyond $\overline{I} = N$, control choices can no more rely on $V(\mathbf{x}, k) < \top$), and invariant violation (e.g., behaviour or an application of the controller outside Inv(q) is not specified).

 $^{^{18}}G$ as a specification is agnostic to when and how guards are used. We use guards as triggers because Algorithm 3 interprets G as a simulator. Updates will be performed as soon as a modal play enters the corresponding guard region.

Algorithm 3 Hybrid game player (with restricted task updates)

1: j	1: procedure HybridPlay(in $(G, \mathbf{U}_{\delta}, T)$; inout $\bar{\mathbf{s}}$)								
2:	$(\mathbf{s}, \rho, k) \leftarrow ((standby, \mathbf{x}_0), \mathbf{p} \in \mathcal{X}_0 _{\mathbf{p}}, 1)$	\triangleright initialise state and goal region							
3:	$\mathbf{repeat} \hspace{0.1in} k{++}$	\triangleright modal play for at most N steps							
4:	switch $\exists e = (q, a, q') \in E \colon \mathbf{x} \in \llbracket \operatorname{grd}(e) \rrbracket \mathbf{d}$	$o \triangleright non-deterministic mode update$							
5:	$(\mathbf{s}, \rho, k) \leftarrow ((q', \operatorname{upd}(e)(\mathbf{x})), \mathbf{p} \in \mathcal{X}_{\mathbf{s}_{a'}^{++}} _{\mathbf{p}})$	1) \triangleright jump (e.g., p -invariant)							
6:	$T \leftarrow \text{UPDTASK}(T)$	\triangleright e.g., change \mathcal{O} to update α							
7:	$G_{\mathbf{s}} \leftarrow (\mathcal{X}_{\mathbf{s}}, \hat{f}, F[\rho])$	\triangleright set scope and winning condition							
8:	$\mathbf{u}_{\mathbf{s}}^* \leftarrow \mathbf{U}_{\delta,N}(G_{\mathbf{s}})$	\triangleright solve bounded modal game							
9:	$\mathbf{d} \leftarrow \text{genDist}(\bar{\mathbf{s}})$ \triangleright	opponent's turn (e.g., random wind)							
10:	$\mathbf{x} \leftarrow \mathbf{x} + \hat{f}(\mathbf{u}_{\mathbf{s}}^*, \mathbf{d}; \mathbf{x}, k)$	\triangleright move and feedback							
11:	until $\Omega \lor \mathbf{x} \notin W(\mathcal{X}_{\mathbf{s}}, k) \lor k \ge N \lor \neg Inv(q)$	\triangleright termination condition							

These three events, by definition, only occur if the current modal game $G_{\mathbf{s}}$ could not be solved. In a simulator, these events can be used for model debugging, while during operational tests, such events may serve, for example, system identification and the adjustment of observers.

Liveness Considerations. Through UPDTASK, Algorithm 3 allows restricted updates of T during operation. To avoid unrealistic moving-target situations, we simplify our setting to a fixed global state space X and route \bar{r} for the entire operation and to a fixed unsafe region α for each modal play. This simplification provides a conservative assumption for achieving strategic liveness. The liveness proofs relying on these assumptions are provided in a working paper [6].

5 Implementation and Experiments

We implemented the Algorithms 1 to 3 in C++ and evaluated their accuracy and performance in four AAV scenarios (i.e., navigating a small domestic yard, a $200 \times 250 m^2$ industrial area, a $400 \times 450 m^2$ neighbourhood with several streets, and a random obstacle cloud) covering a range of realistic situations. With our simulator (Algorithm 3), we illustrate plays of the hybrid game (Figure 4a) for the yard and industrial scenarios from [11] as well as the more complicated streets scenario. We also highlight some data (Table 1).

Aerial Delivery Game Parameters. For the mentioned scenarios, we use

$$\rho \equiv \mathbf{p} \in \mathcal{X}_{\mathbf{s}_{\mathcal{I}}^{++}}|_{\mathbf{p}} \qquad \alpha \equiv \operatorname{coll}_{\mathsf{f}}(\mathcal{O}) \qquad \text{and} \qquad \lambda(\mathbf{u}, \mathbf{d}; \mathbf{s}, k) = \mathbf{x}^{2} + \mathbf{u}^{2}.$$
(7)

Informally, ρ enforces the AAV to reach the next waypoint segment $(\mathbf{p}_i, \mathbf{p}_{i+1})$, except for the elevate segment during departure, and α enforces the vehicle to avoid static obstacles. A corresponding collision check (i.e., safety pre-shielding) with \mathcal{O} is performed for each state and pair of control and disturbance inputs in Line 7 of Algorithm 1 and cached for all time steps when α is computed for L.



Figure 5: Play of G with $\bar{r} = {\mathbf{p}_i}_{i=0}^n$ for the industrial scenario (n = 13)

In further experiments not shown here, we worked with a non-linear variant of λ reciprocally weighing-in the distance to the nearest obstacle.

In Algorithm 2, we initialised our settings with a fixed I defining N to be the maximum horizon and $\delta_I = 0$ to disable temporal scope extension. For $\Delta_{\mathcal{X}}$, we employ a fixed small symmetric padding of 2 units across all scenarios. Overall, **U** is only used for $q \in \{\text{depart, cruise, arrive}\}$ where $\hat{f} = fly$. In Algorithm 3, we use $\Omega \equiv q = \text{standby}$. Under δ -perforation (1), UPDTASK is constrained to liveness-preserving updates of \mathcal{O} .

Note how Algorithm 3 leaves standby, which fulfils Ω , by immediately performing the globally enabled start event. In the AAV example, we keep jumps deterministic. Our implementation would resolve non-deterministic jumps by taking the first available. However, in complex applications, the discrete part of the hybrid game will require an informed strategy as well.

To focus on the more relevant cases, GENDIST in Line 8 only generates horizontal disturbances. In particular, wind (N, W, S, E) is simulated randomly, following a semi-Markov scheme with finite memory. In the scenarios, wind is more likely to change gradually than spontaneously.

Hybrid Game Plays. For validating our notion of robustness, the Figures 5 and 6 illustrate the plays of G in the industrial and streets scenarios and Figure 7 provides a 3D walkthrough of a play in the streets scenario. A play for the yard scenario is shown in Figure 1. The blue AAV trajectory marks the center of a reference tube (i.e., $\mathbf{d} = \mathbf{0}$) and, for comparison, the red trajectory is the result of random disturbance being applied (i.e., randomised \mathbf{d} no worse than \mathbf{d}^*). We apply the cost function from (7).

The rectangle around a segment $(\mathbf{p}_{i+1}, \mathbf{p}_{i+2})$ indicates the goal region ρ of the route tracking task for segment $(\mathbf{p}_i, \mathbf{p}_{i+1})$. ρ circumscribes the next pair of waypoints and, thus, allows edge cases (cf. Figure 1) where certain waypoints (e.g., \mathbf{p}_5) are circumvented for the benefit of shortcuts (e.g., $(\mathbf{p}_4, \mathbf{p}_6)$).

3D visualisation can provide valuable insights into the vertical scene topol-



Figure 6: Play of G with $\bar{r}=\{\mathbf{p}_i\}_{i=0}^n$ for the streets scenario (n=11)



Figure 7: Snapshots of a robust play in the *streets* scenario. In blue, the nearoptimal reference trajectory enforced by \mathbf{u}^* without disturbance being applied. In red, a simulation of \mathbf{u}^* with random **d** applied. Transparent rectangles visualise goal regions.

Scenario	$\begin{array}{c} \text{Play} \\ \text{time} \\ t(\bar{r}) \end{array}$	$\begin{array}{c} {\rm Max.} \\ \# \\ {\rm stages} \end{array}$	Max. mem. usage	$\max_{ X}$	Avg. $ X $	$\begin{array}{c} \text{Max.} \\ t \text{ per} \\ \text{dDP} \end{array}$	Avg. t per 10^6	$\frac{\text{Avg.}}{\frac{t(\mathbf{p}_0)}{t(\mathbf{p}_{>0})}}$
	[h:m]	(N)	[GiB]	$\cdot 10^{6}$	$\cdot 10^{6}$	[m:s]	states [s]	
Yard	01:39	30	20.1	105	36.4	25:38	0.597	1.94
Industrial	01:51	60	37.4	108	25.8	47:58	0.798	3.52
Random	01:03	30	5.8	65	25.7	06:54	0.626	1.56
Streets	06:43	50	36.2	172	48.7	51:11	1.110	4.50

Table 1: δ -robust synthesis compared by metric across the scenarios

ogy and the trajectory resulting from a play. During model validation and game design, it can help to identify parameters, such as beneficial placement of waypoints in the context of scope shaping or the robustness margin δ . The 3D scenery can increase the confidence in the robustness guarantee provided by the synthesised controllers through a comparison of the trajectories resulting from a worst-case play and a play with random disturbance.

Data from the Plays. Table 1 summarises key indicators of the scenarios, such as the total run-time $t(\bar{r})$ of the play for route \bar{r} , time $t(\mathbf{p}_i)$ to compute \mathbf{u}^* for segment $(\mathbf{p}_i, \mathbf{p}_{i+1})$, and the maximum number (#) of states in \mathcal{X} . Moreover, the expected savings in peak memory usage of the quasi-stationary controller are summarised in Figure 8 for comparison.

6 Evaluation and Discussion

Performance. Our example uses a linear approximation and a quadratic cost function and is, thus, amenable to a solution by quadratic programming. However, keeping our approach more widely applicable requires significantly more time and memory than (non-linear) model-predictive control (MPC) schemes (Table 1). Clearly, the C++ prototype is not ready for use in real-time settings. Nevertheless, selective state discretisation and omission (e.g., in-



Figure 8: Peak memory usage of Algorithm 3 by scenario and controller type

terpolating across state-time dimensions, improved parameter settings) and high-performance parallelisation can reduce average DDP run-time and space requirements by at least two orders of magnitude, loosing optimality and precision but keeping much of the exhaustiveness of the scenario coverage. Such a run-time reduction increases segment tracking speed as well as freedom in route

planning.

Operationalisation of the Hyper-Policy. Lines 8 to 9 of Algorithm 3 represent the environment part of the control loop: Based on the observation (i.e., \mathbf{x} is an estimate provided by an observer), control input and disturbance are applied in parallel, followed by the next observation. Because of this real-time dependency, Lines 6 to 7 need to be pre-computed to be available after the jumps.

Ideally, scenario parameters allow enough time to compute \mathbf{u} for the next segment $(\mathbf{p}_i, \mathbf{p}_{i+1})$ based on data from local environmental perception before reaching \mathbf{p}_i . Alternatively, there is time to compute \mathbf{u} for $(\mathbf{p}, \mathbf{p}_i)$ before being required to give up \mathbf{p} as a potential waiting position (e.g., where \mathbf{u} is a PID controller to robustly hold the AAV at \mathbf{p}). However, if the scenario at hand does not permit a strong enough approximation, \mathbf{U} will have to be pre-computed (prior to or during flight)¹⁹ for a sufficiently long prefix of \bar{r} .

Sound Value-Fixpoint Approximations. In the Algorithms 1 and 2, the fixpoint V^* , required for \mathbf{u}^* to be stationary (i.e., applicable independent of time, $N \approx \infty$, [21]) and optimal, is approximated. In particular, \hat{fp}_{dDP} replaces the ideal but more expensive, non-simultaneous fixpoint check $V(\mathcal{X}_{\mathbf{s}}, k) = V(\mathcal{X}_{\mathbf{s}}, k+1)$ for each k in Line 9 of Algorithm 1. On the one hand, the ideal check might, due to numerical imprecision and instability or a too small N, never be successful. On the other hand, the faster check $\top \notin V(\mathbf{x} \oplus \Delta_{\delta}, k)$ in Line 7 of Algorithm 2 poorly approximates V^* and would turn our synthesis into plain shortest-path finding. The latter might, however, suffice in some applications.

Accepting some deviation from optimality, we instead use $|W(\mathcal{X}_{\mathbf{s}}, k)| = |W(\mathcal{X}_{\mathbf{s}}, k+1)|$ to check in Algorithm 1 whether the number of states stabilises, where a robustly safe and goal-reaching trajectory for at least k steps exists. Then, we perform $\top \notin V(\mathbf{x} \oplus \Delta_{\delta}, k)$ in Algorithm 2. Although $W(\mathcal{X}_{\mathbf{s}}, k)$ may still evolve while preserving its cardinality, Algorithm 3 then either ensures $\mathbf{x} \in W(\mathcal{X}_{\mathbf{s}}, k)$ before leaving the solver in Line 7 or it terminates with a failure because of $W(\mathcal{X}_{\mathbf{s}}, k) = \emptyset$ in Line 10.

A comparison of the non-stationary \mathbf{u}^* , obtained for [k, N] by the above termination rule, and the quasi-stationary $\mathbf{u}^\infty \colon \mathbb{X} \to \mathcal{U}$, derived via $\mathbf{u}^\infty(\mathbf{x}) =$ $\mathbf{u}^*(\mathbf{x}, k)$, indicates (expected) improvements of \mathbf{u}^∞ (green) over \mathbf{u}^* (blue) in the undisturbed case (Figure 9). Moreover, as the environment follows the controller (i.e., \mathbf{u}^* cannot change its current choice based on **d**'s current choice), applying disturbance (red) will have the same effect on \mathbf{u}^∞ . Figure 9 (top right) illustrates how \mathbf{u}^∞ deviates from \mathbf{u}^* to get closer to optimality according to (5).

Generalisation. Horizon N, stage k, and alphabets \mathcal{U} and \mathcal{D} are largely left implicit in J, V, W, \hat{f} , and \mathbf{x}^{ud} . Also, we employ time resolution 1. However, making these parameters explicit leads to non-essential extensions and our approach can be parameterised by N and enhanced to time-varying dynamics

¹⁹Pre-computation is feasible for scope-padded segments $(\mathbf{p}_i, \mathbf{p}_{i+1})$ but not arbitrary s.



Figure 9: Partial trajectory triples (non-stationary reference, random disturbance applied, quasi-stationary reference) from the industrial (top left), random (top right), streets (bottom left), and yard (bottom right) scenarios. The random scenario highlights differences between non- and quasi-stationary references.

and alphabets \mathcal{U}_k and \mathcal{D}_k as well as non-unit time resolutions. Moreover, it can be extended from linear approximation (Algorithm 1 Line 6) to non-linear approximation of the dynamics and to using non-linear cost-functions.

7 Conclusion

In this work, we specialise a conventional controller synthesis algorithm for a parametric discretised variant of a weighted hybrid game G. We provide a model for the integrated assurance of robust safety, liveness, and near-optimality of controllers for G that are synthesised online, that is, during G's execution. The abstraction we chose combines hybrid games (e.g., reach-avoid reasoning), performance optimisation, and numerical aspects (e.g., data sampling, quantisation) of digital control in a way amenable to formal reasoning. This combination enables us to reason about robust safety and liveness guarantees of controllers and how these guarantees impact the assurance of an overall system. Proofs of key statements used in this work (e.g., Algorithm 3 ensures that $\mathbf{x} \in W(\mathcal{X}_{\mathbf{s}}, k)$; necessary and sufficient conditions for solvability of the weighted hybrid game) are discussed in a companion working paper [6].

Our application focuses on the modelling of AAVs, the online synthesis of tactical controllers, and integrating the latter with strategic and supervisory control. Our approach can be an alternative if a combination of RL with a shielding scheme is not desirable (e.g., lack of generalisability, controllability, or explainability of RL's value function approximation).

In future work, we will extend our approach to synthesise more complex strategies and refine the integration with our work [7] at the supervisory control level. We will improve the performance of our algorithms as suggested in Section 6 and extend our model to be able to deal with multiple AAVs that can form an intelligent aerial transport collective.

References

- Althoff, M., Dolan, J.M.: Online verification of automated road vehicles using reachability analysis. IEEE Trans. Rob. 30(4), 903–918 (2014). https://doi.org/10.1109/TRO.2014.2312453
- [2] Bertram, J., Wei, P., Zambreno, J.: A fast markov decision process-based algorithm for collision avoidance in urban air mobility. IEEE Trans. Intell. Transp. Syst. 23(9), 15420–15433 (2022). https://doi.org/10.1109/ tits.2022.3140724
- [3] Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. 1. Athena Scientific, 4th edn. (2017)
- [4] Clement, E., Perrin-Gilbert, N., Schlehuber-Caissier, P.: Layered Controller Synthesis for Dynamic Multi-agent Systems, pp. 50–68. Springer, Cham, CH (2023). https://doi.org/10.1007/978-3-031-42626-1_4
- [5] Gleirscher, M.: Supervision of intelligent systems: An overview. In: Applicable Formal Methods for Safe Industrial Products Essays Dedicated to Jan Peleska on the Occasion of His 65th Birthday, LNCS, vol. 14165, pp. 1–21. Springer, Cham, CH (2023). https://doi.org/10.1007/978-3-031-40132-9_13
- [6] Gleirscher, M.: Solvability of approximate reach-avoid games. CoRR (2025). https://doi.org/10.48550/arXiv.2502.04544
- [7] Gleirscher, M., Calinescu, R., Douthwaite, J., Lesage, B., Paterson, C., Aitken, J., Alexander, R., Law, J.: Verified synthesis of optimal safety controllers for human-robot collaboration. Sci. Comput. Program. 218, 102809 (2022). https://doi.org/10.1016/j.scico.2022.102809
- [8] Gu, R., Jensen, P.G., Poulsen, D.B., Seceleanu, C., Enoiu, E., Lundqvist, K.: Verifiable strategy synthesis for multiple autonomous agents: a scalable approach. Int. J. Softw. Tools Technol. Trans. (2022). https://doi.org/ 10.1007/s10009-022-00657-z
- [9] Henzinger, T.A.: The theory of hybrid automata. In: Verification of Digital and Hybrid Systems, NATO ASI Series F: Computer and Systems Sciences, vol. 170, pp. 265–92. Springer (2000). https://doi.org/10.1007/ 978-3-642-59615-5_13

- [10] Heuillet, A., Couthouis, F., Díaz-Rodríguez, N.: Explainability in deep reinforcement learning. Knowledge-Based Syst. 214, 106685 (2021). https: //doi.org/10.1016/j.knosys.2020.106685
- [11] Hönnecke, P.: Constrained Hybrid Optimal Control of Aerial Transport Systems. Master thesis, under sup. of M. Gleirscher, UBremen (2024)
- [12] Ivanov, R., Jothimurugan, K., Hsu, S., Vaidya, S., Alur, R., Bastani, O.: Compositional learning and verification of neural network controllers. ACM Trans. Embed. Comput. Syst. 20(5s), 1–26 (2021). https://doi.org/10. 1145/3477023
- [13] Kanashima, K., Ushio, T.: Finite-horizon shield for path planning ensuring safety/co-safety specifications and security policies. IEEE Access 11, 11766-11780 (2023). https://doi.org/10.1109/access.2023.3241946
- [14] Lewis, F.L., Vrabie, D., Vamvoudakis, K.G.: Reinforcement learning and feedback control. IEEE Control Syst. 32(6), 76–105 (2012). https://doi. org/10.1109/mcs.2012.2214134
- [15] Li, N., Goubault, E., Pautet, L., Putot, S.: A real-time NMPC controller for autonomous vehicle racing. In: ICACR. pp. 148–155. IEEE (2022). https://doi.org/10.1109/icacr55854.2022.9935523
- [16] Mitsch, S., Ghorbal, K., Vogelbacher, D., Platzer, A.: Formal verification of obstacle avoidance and navigation of ground robots. Int. J. Rob. Res. 36(12), 1312–1340 (2017). https://doi.org/10.1177/0278364917733549
- Schnittka, T., Gleirscher, M.: Synthesising robust strategies for robot collectives with recurrent tasks: A case study. In: Luckcuck, M., Xu, M. (eds.) FM Auton. Sys. (FMAS), 6th Workshop. EPTCS, vol. 411, pp. 109–125. OPA (2024). https://doi.org/10.4204/EPTCS.411.7
- [18] Shalev-Shwartz, S., Shammah, S., Shashua, A.: Safe, multi-agent, reinforcement learning for autonomous driving. Tech. rep., Mobileye (2016)
- [19] Taye, A.G., Valenti, R., Rajhans, A., Mavrommati, A., Mosterman, P.J., Wei, P.: Safe and scalable real-time trajectory planning framework for urban air mobility. J. Aero. Inf. Sys. pp. 1–10 (2024). https://doi.org/ 10.2514/1.i011381
- [20] Thumm, J., Althoff, M.: Provably safe deep reinforcement learning for robotic manipulation in human environments. In: ICRA. IEEE (2022). https://doi.org/10.1109/icra46639.2022.9811698
- [21] Tomlin, C.J., Lygeros, J., Sastry, S.S.: A game theoretic approach to controller design for hybrid systems. Proc. IEEE 88(7), 949-970 (2000). https://doi.org/10.1109/5.871303