Safe multi-agent motion planning via filtered reinforcement learning

Vinod, Abraham P.; Safaoui, Sleiman; Chakrabarty, Ankush; Quirynen, Rien; Yoshikawa, Nobuyuki; Di Cairano, Stefano

TR2022-053 May 24, 2022

Abstract

We study the problem of safe multi-agent motion planning in cluttered environments. Existing multi-agent reinforcement learning-based motion planners only provide approximate safety enforcement. We propose a safe reinforcement learning algorithm that leverages single-agent reinforcement learning for target regulation and a subsequent convex optimization-based filtering that ensures the collective safety of the system. Our approach yields a safe, real-time implementable multi-agent motion planner that is simpler to train and enforces safety as hard constraints. Our approach can handle state and control constraints on the agents, and enforce collision avoidance among themselves and with static obstacles in the environment. Numerical simulations and hardware experiments show the efficacy of the approach.

IEEE International Conference on Robotics and Automation (ICRA) 2022

Safe multi-agent motion planning via filtered reinforcement learning

Abraham P. Vinod*, Sleiman Safaoui, Ankush Chakrabarty, Rien Quirynen,
Nobuyuki Yoshikawa, and Stefano Di Cairano

***Abstract*— We study the problem of safe multi-agent motion planning in cluttered environments. Existing multi-agent reinforcement learning-based motion planners only provide approximate safety enforcement. We propose a safe reinforcement learning algorithm that leverages single-agent reinforcement learning for target regulation and a subsequent convex optimization-based filtering that ensures the collective safety of the system. Our approach yields a safe, real-time implementable multi-agent motion planner that is simpler to train and enforces safety as hard constraints. Our approach can handle state and control constraints on the agents, and enforce collision avoidance among themselves and with static obstacles in the environment. Numerical simulations and hardware experiments show the efficacy of the approach.**

I. Introduction

Multi-agent motion planning in cluttered environments is an essential component of safe autonomy for transportation, logistics, precision agriculture, search and rescue operations, and monitoring. In all of these applications, we need real-time implementable algorithms that can simultaneously plan trajectories for multiple agents to their respective targets and adapt to changing environments using collected data, while ensuring collision avoidance with static obstacles and other agents moving in the environment.

Researchers have proposed several approaches to address the multi-agent motion planning problem, drawing from optimization, robotics, control, and learning. Some of these approaches include mixed-integer programming [1], adaptive roadmaps [2], buffered Voronoi cells [3], sequential convex programming [4], and barrier certificates [5]. Specialized algorithms to tackle certain aspects of the planning problem like collision avoidance have also been proposed [4], [6]. A common limitation among these planning algorithms is that they do not leverage data collected from past trajectory executions in order to improve future motion plans. Learning-based approaches, especially using reinforcement learning (RL) [7]–[11], have been designed to systematically exploit prior data, but in most cases incorporate safety via soft constraints for computational tractability, thereby discouraging, yet not eliminating the risk of potentially unsafe operation. Additionally, multi-agent RL algorithms face the challenge of non-stationarity of the environment arising from concurrent learners [12].

The problem of ensuring safety in RL-based motion planning has gained more attention in recent years [13]–[15]. Among these works, the closest work to the proposed approach is that in [15]. Here, a two-player game is solved offline to synthesize a "shield" that ensures that the actions of the RL-based controllers are congruent with the safety objectives of the problem. Due to the inherent computational limitations in solving two-player games, the approach in [15] is limited to problems with discrete state and action space. In contrast, we propose an optimization-based safety filter that can accommodate continuous state and action spaces for stabilizable linear dynamics and is real-time implementable.

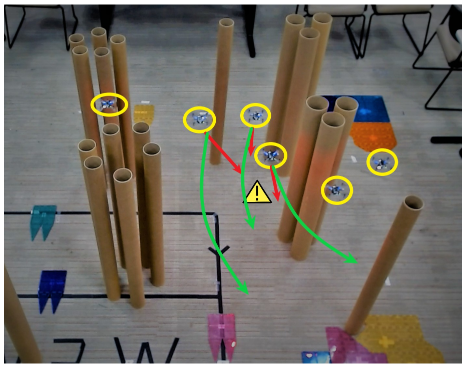


Fig. 1. Existing reinforcement learning-based motion planners (red arrows) treats safety as soft constraints, which is undesirable in safety-critical applications. We propose a quadratic program-based safety filter that renders such motion planners safe (long green arrows) by enforcing safety as hard constraints. See supplementary material for the video of the hardware experiments using CrazyFlies.

The main contribution of this work is *a safe multi-agent motion planner that enforces safety as hard constraints in planning while leveraging prior data.* Specifically, we develop a quadratic programming-based safety filter that ensures collective safety of a multi-agent system, while minimizing the deviation from the control input prescribed by a RL-based motion planner. Here, *collective safety* refers to avoidance of collision between agents and between agents and obstacles, and satisfaction of constraints on the states and inputs of the agents at all times. Our approach can accommodate agents with any stabilizable linear dynamics, collision avoidance constraints (typically non-convex state constraints), convex input constraints, and is real-time implementable. We demonstrate the proposed motion planner in simultaneous navigation of six Crazyflie quadrotors to their desired target locations in a crowded static environment.

II. Problem formulation and preliminaries

Notation: 0_d denotes the zero vector in $\mathbb{R}^d$, I_d denotes the d-dimensional identity matrix, $\mathbb{N}_{[a,b]}$ denotes the subset

* Corresponding author. Email: `vinod@merl.com`.
A. Vinod, S. Safaoui, A. Chakrabarty, R. Quirynen, and S. Di Cairano are with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA. N. Yoshikawa is with Mitsubishi Electric Corporation, Japan.

of natural numbers between (and including) $a, b \in \mathbb{N}$, $a \leq b$, and $\oplus, \ominus$ denotes Minkowski sum and difference respectively. We denote the support function of a convex and compact set $\mathcal{A}$ by $S_{\mathcal{A}}(\ell) \triangleq \sup_{x \in \mathcal{A}} \ell \cdot x$ for any $\ell \in \mathbb{R}^d$ [16]. We denote continuous time instants with $\tau, t \in \mathbb{R}$, $\tau, t \geq 0$, and discrete time sampling instants with $k \in \mathbb{N}$.

A. Problem setup

Consider $N_A \in \mathbb{N}$ homogeneous agents with the following stabilizable, continuous-time linear dynamics,

$$\dot{x}_i = Ax_i + Bu_i, \tag{1}$$

with agent state $x_i \in \mathbb{R}^n$ and input $u_i \in \mathcal{U} \subseteq \mathbb{R}^m$ for each $i \in \mathbb{N}_{[1,N_A]}$, and $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$. We define performance objectives and safety constraints based on the position $p_i(t)$ of the agents,

$$p_i(t) = Cx_i(t) \in \mathbb{R}^d, \tag{2}$$

where $C \in \mathbb{R}^{d \times n}$. We also assume that the number of inputs m is no smaller than the dimension of the position vector d, and that the admissible input set $\mathcal{U}$ is a convex polytope. Additionally, for the ease of exposition, we assume that the agents have identical convex and compact rigid bodies, denoted by $\mathcal{A} \subset \mathbb{R}^d$, such that $0_d \in \mathcal{A}$.

We represent the bounds on the environment using a polytopic set $\mathcal{K}$. For some $N_{\mathcal{K}} \in \mathbb{N}$ and $\{h_i, g_i\}_{i=1}^{N_{\mathcal{K}}}$ with $h_i \in \mathbb{R}^d$ and $g_i \in \mathbb{R}$, $\mathcal{K} = \bigcap_{i \in \mathbb{N}_{[1,N_{\mathcal{K}}]}} \{p \in \mathbb{R}^d : h_i \cdot p \leq g_i\}$. The obstacles in the environment are located at $c_j \in \mathbb{R}^d$ with convex and compact rigid bodies $\mathcal{O}_j \subset \mathbb{R}^d$, with $0_d \in \mathcal{O}_j$.

We seek to drive the agents towards their respective target positions $q_i \in \mathbb{R}^d$. Additionally, we require that the agents are *collectively safe* at all times, as formalized below.

Definition 1 (Collective safety). *The agents are* collectively safe *at time t, if all of the following criteria are met:*

1) Keep-in constraints*: We require $p_i(t) \oplus \mathcal{A} \subseteq \mathcal{K}$.*
2) Static obstacle avoidance constraints*: We require the agents to avoid $N_O \in \mathbb{N}$ static obstacles present in the environment. The i^{th} agent and the j^{th} obstacle are in collision when $(p_i(t) \oplus \mathcal{A}) \cap (c_j \oplus \mathcal{O}_j)$ is non-empty.*
3) Inter-agent collision avoidance*: We require that all agents avoid collision among themselves. Agents $i, j \in \mathbb{N}_{[1,N_A]}$ are in collision, when $(p_i(t) \oplus \mathcal{A}) \cap (p_j(t) \oplus \mathcal{A})$ is non-empty.*

Problem 1 (Stabilizable linear dynamics). *Design a multi-agent motion planner that navigates the agents* (1) *to their respective targets $\{q_i\}_{i=1}^{N_A}$ such that the agents are collectively safe at discrete times $k\delta$, $k \in \mathbb{N}$, for some user-specified sampling time δ.*

While the proposed solution to Problem 1 can handle a broad class of dynamical systems, the safety properties of the approach are limited to user-specified discrete-time steps due to computational limitations. We can recover the continuous-time safety by specializing the proposed approach for double integrator-based dynamics. In this case, for the ease of exposition, we focus on motion planning problems over two-dimensional space ($d = 2$), with double integrator dynamics for each of the dimensions. Specifically, we describe the dynamics of the agents with states $x(t) = [p(t)\ v(t)] \in \mathbb{R}^4$ (two-dimensional position and velocity),

$$A = \begin{bmatrix} 0_2 & I_2 \\ 0_2 & 0_2 \end{bmatrix},\ B = \begin{bmatrix} 0_2 \\ I_2 \end{bmatrix},\ C = \begin{bmatrix} I_2 & 0_2 \end{bmatrix}. \tag{3}$$

Several robotic platforms including quadrotors are typically controlled via a combination of a high-level planner that abstracts the dynamics using double integrator-based dynamics, and a low-level controller for trajectory tracking realizing such dynamics [4].

Problem 2 (Double integrator dynamics). *Design a multi-agent motion planner that navigates the agents with dynamics* (3) *to their respective targets $\{q_i\}_{i=1}^{N_A}$ such that the agents are collectively safe at all times $t \geq 0$.*

B. Reinforcement learning for single-agent motion planning

We briefly discuss the design of a RL-based motion planner for a single agent with linear dynamics (1) that can avoid static obstacles. First, we discretize (1) in time using a zero-order hold with a user-specified sampling time $\Delta > 0$,

$$x(k+1) = A_\Delta x(k) + B_\Delta u(k). \tag{4}$$

Here, $A_\Delta = e^{A\Delta} \in \mathbb{R}^{n \times n}$ and $B_\Delta = \int_0^\Delta e^{A(\Delta - t)} B dt \in \mathbb{R}^{n \times m}$ are functions of Δ. We define a low-level controller for (4) that is held constant between sampling times,

$$u^{\text{RL}}(k) \triangleq Kx(k) + Fr(k). \tag{5}$$

Here, K is a stabilizing gain matrix (i.e., all eigenvalues of $(A + BK)$ have magnitudes smaller than 1), $F \in \mathbb{R}^{m \times d}$ satisfies $C(I - (A+BK))^{-1} BF = I_d$, and $r(k)$ is a discrete-time command position set by the RL-based motion planner. From (4) and (5),

$$x(k+1) = (A_\Delta + B_\Delta K)x(k) + B_\Delta F r(k). \tag{6}$$

For any constant reference position $r(k) = r \in \mathbb{R}^d$, the closed-loop dynamics (6) guarantees $\lim_{k \to \infty} p(k) = r$ [17].

We consider the following deterministic Markov decision process for an agent with linear dynamics (6) with a pre-specified target position $q \in \mathbb{R}^d$:

1) *Observation space*: We define the observation vector $o(k) \in \mathbb{R}^{n+d+N_O d}$ as the concatenated vector containing $x(k) \in \mathbb{R}^n$, the displacement of the agent's current position to the target $(p(k) - q) \in \mathbb{R}^d$ and to the N_O static obstacles $(p(k) - c_j) \in \mathbb{R}^d$, for all $j \in \mathbb{N}_{[1,N_O]}$.
2) *Action space*: The action $a(k) \in \mathcal{A} \subset \mathbb{R}^d$ determines the reference position as a bounded perturbation $a(k)$ to the target q, $r(k) = q + a(k)$.
3) *Step function*: We obtain the next state $x(k+1)$ by (6).
4) *Reward function*: The instantaneous reward function is

$$R(p(k)) = \alpha_{\text{obs}} \sum_{j=1}^{N_O} \frac{1}{\|p(k) - c_j\|^2 - \gamma_j^2} + \alpha_{\text{tgt}} \|p(k) - q\|$$

with reward parameters $\alpha_{\text{obs}}, \alpha_{\text{tgt}} \leq 0$ and $\gamma_j \geq 0$. Here, γ_j is the radius of the smallest volume origin-centered ball that covers the set $\mathcal{O}_j \oplus (-\mathcal{A})$ for each $j \in \mathbb{N}_{[1,N_O]}$. We terminate an episode (denoted by $k = \infty$) when the agent either reaches the target, violates the keep-in constraints, or collides with a static obstacle with a terminal reward or penalty,

$$R(p(\infty)) = \begin{cases} R_{\text{target}}, & \text{if } \|p(\infty) - q\| \leq d \ , \\ P_{\text{keep-in}}, & \text{if } p(\infty) \oplus \mathcal{A} \not\subseteq \mathcal{K}, \\ P_{\text{obstacle}}, & \text{if agent hits an obstacle,} \end{cases}$$

with $R_{\text{target}} \geq 0$ and $P_{\text{keep-in}}, P_{\text{obstacle}} \leq 0$.

In conjunction with the low-level controller $u_i^{\text{RL}}(k)$, a policy that solves the Markov decision process can drive an agent starting at an arbitrary safe position within the keep-in set towards q without colliding with any of the static obstacles. There are several algorithms to solve such Markov decision problems, along with computational packages that can synthesize policies based on neural networks [18], [19].

III. Safe multi-agent motion planning using learning and optimization

The general framework of the proposed solution to address Problems 1 and 2 is illustrated in Figure 2. We design the motion planner in two stages. First, every agent computes a low-level controller $u_i^{\text{RL}}(k)$ based on their respective target and the corresponding single-agent motion planners. Next, a centralized quadratic program-based safety filter generates $u_i^{\text{safe}}(k)$ that renders the overall system collectively safe, while minimizing the deviations from their respective learning-based controllers. Thus, we assign the task of regulating towards the target to the learning-based controller and the task of inter-agent collision avoidance to the safety filter. Both of the modules share the remaining tasks of ensuring static obstacle avoidance and satisfaction of the state and control constraints like the keep-in constraints.

Our approach has several advantages over plain RL-based motion planners: First, it utilizes convex optimization to enforce safety as hard constraints ensuring real-time implementability. Second, it significantly reduces the dimensionality of the learning problem since the policy is synthesized for a single agent, which further simplifies the training process. Finally, the use of a shared policy in our approach also avoids the need for concurrent learning, typically present in multi-agent reinforcement learning. Concurrent learning requires the learning agent to account for how the other agents behave, which invalidates the stationary Markovian property typically used in the convergence proofs of RL algorithms [12]. Concurrent learning is also known to cause significant instability to training algorithms [11], [12].

A. Safety filter for stabilizable linear dynamics: Setup

Our objective is to ensure collective safety of the overall multi-agent system for a user-specified continuous-time safety horizon $T_{\text{safe}} \in \mathbb{R}, T_{\text{safe}} > 0$. Define $\mathbb{T}_{\text{safe}} = [0, T_{\text{safe}}]$. At each discrete-time step $k \in \mathbb{N}$, we compute $u_i^{\text{safe}}(k)$ for

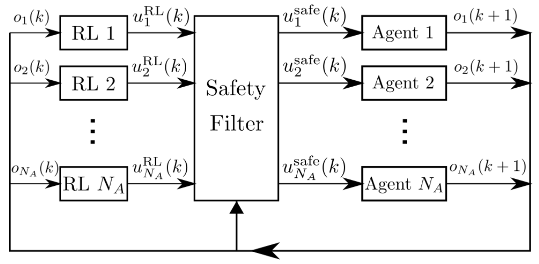


Fig. 2. The proposed solution combines single-agent RL-based motion planning with a safety filter for safe multi-agent motion planning.

each agent $i \in \mathbb{N}_{[1,N_A]}$ that is held constant over the time interval $\{k\Delta\} \oplus \mathbb{T}_{\text{safe}}$, while ensuring 1) collectively safe operation during the time interval, 2) minimal deviation of $u_i^{\text{safe}}(k)$ from $u_i^{\text{RL}}(k)$, and 3) satisfaction of actuation limits. We solve the following optimization problem at each k,

$$\underset{u_1^{\text{safe}}(k),\ldots,u_{N_A}^{\text{safe}}(k)}{\text{minimize}} \quad \sum_{i \in \mathbb{N}_{[1,N_A]}} \lambda_i \|u_i^{\text{RL}}(k) - u_i^{\text{safe}}(k)\|_2^2 \tag{7a}$$

$$\text{subject to} \quad u_1^{\text{safe}}(k), \ldots, u_{N_A}^{\text{safe}}(k) \in \mathcal{U} \tag{7b}$$

$$\left.\begin{matrix} i \in \mathbb{N}_{[1,N_A]}, \\ t \in \{k\Delta\} \oplus \mathbb{T}_{\text{safe}}, \end{matrix}\right\} \quad p_i(t|k) = C\left(\mathscr{A}_{t|k} x_i(k) + \mathscr{B}_{t|k} u_i^{\text{safe}}(k)\right), \tag{7c}$$

$$t \in \{k\Delta\} \oplus \mathbb{T}_{\text{safe}}, \quad \text{All agents are collectively safe at } t, \tag{7d}$$

with user-specified weighting parameters $\lambda_i \geq 0$, $\mathscr{A}_{t|k} = e^{A(t-k\Delta)} \in \mathbb{R}^{n \times m}$ and $\mathscr{B}_{t|k} = \int_0^\Delta e^{A(t-k\Delta-s)} B ds$. The cost (7a) provides a weighted sum of the squared-deviations between the low-level controllers. The convex constraint (7b) enforces the control bounds. The dynamics (7c) provides the agent positions $p_i(t|k)$ at time $t \in \{k\Delta\} \oplus \mathbb{T}_{\text{safe}}$, given their states at a continuous time instant $k\Delta$.

We now briefly discuss constraints for (convex) enforcement of collective safety (7d).

1) Keep-in constraints: The keep-in constraint is equivalent to $p_i(t|k) \in \mathcal{K} \ominus \mathcal{A}$ by the definition of Minkowski difference, which is easy to compute for a polytopic $\mathcal{K}$ [16],

$$\mathcal{K} \ominus \mathcal{A} = \cap_{i \in \mathbb{N}_{[1,N_{\mathcal{K}}]}} \{p : h_i \cdot p \leq g_i - S_{\mathcal{A}}(h_i)\}\,. \tag{8}$$

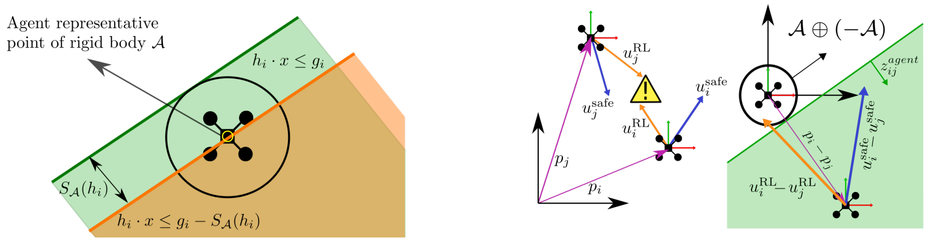
Figure 3 illustrates the (linear) keep-in constraints.

2) Static obstacle avoidance constraints: Agent $i \in \mathbb{N}_{[1,N_A]}$ must avoid collision with a static obstacle $j \in \mathbb{N}_{[1,N_O]}$ located at c_j at times $t \in \{k\Delta\} \oplus \mathbb{T}_{\text{safe}}$. Using computational geometry arguments, we have

$$\begin{aligned} &(\{p_i(t|k)\} \oplus \mathcal{A}) \cap (\{c_j\} \oplus \mathcal{O}) = \emptyset \\ &\iff \nexists y_O \in \mathcal{O}_j, \nexists y_A \in \mathcal{A},\ p_i(t|k) - c_j = y_O - y_A \\ &\iff \{p_i(t|k) - c_j\} \not\in \mathcal{O}_j \oplus (-\mathcal{A}), \end{aligned} \tag{9}$$

for every $i \in \mathbb{N}_{[1,N_A]}$, $j \in \mathbb{N}_{[1,N_O]}$, and $t \in \{k\Delta\} \oplus \mathbb{T}_{\text{safe}}$. Since the set $\mathcal{O}_j \oplus (-\mathcal{A})$ is convex, (9) is a non-convex constraint. We convexify (9) via a separating hyperplane for the point $\{p_i(t|k) - c_j\}$ and the convex set $\mathcal{O} \oplus (-\mathcal{A})$ to obtain linear constraints in $u_i^{\text{safe}}(k)$. Specifically, for any choice of a unit vector $z_{ij}^{\text{obs}} \in \mathbb{R}^d$, $\|z_{ij}^{\text{obs}}\| = 1$ [20],

$$z_{ij}^{\text{obs}} \cdot (p_i(t|k) - c_j) \geq S_{\mathcal{O}_j}(z_{ij}^{\text{obs}}) + S_{(-\mathcal{A})}(z_{ij}^{\text{obs}}) \implies (9).$$



Agent representative point of rigid body $\mathcal{A}$
$h_i \cdot x \leq g_i$
$S_{\mathcal{A}}(h_i)$
$h_i \cdot x \leq g_i - S_{\mathcal{A}}(h_i)$
u_j^{RL}
u_j^{safe}
u_i^{safe}
u_i^{RL}
p_j
p_i
$\mathcal{A} \oplus (-\mathcal{A})$
z_{ij}^{agent}
$p_i - p_j$
$u_i^{\text{safe}} - u_j^{\text{safe}}$
$u_i^{\text{RL}} - u_j^{\text{RL}}$

Fig. 3. Collective safety constraints (Definition 1) enforced as linear constraints — (left) keep-in constraints, and (right) inter-agent collision avoidance.

3) Inter-agent collision avoidance constraints: Similarly to (9), inter-agent collision avoidance can be enforced as a linear constraint,

$$\begin{aligned}&(\{p_i(t|k)\} \oplus \mathcal{A}) \cap (\{p_j(t|k)\} \oplus \mathcal{A}) = \emptyset\\ \iff &\{p_i(t|k) - p_j(t|k)\} \notin \mathcal{A} \oplus (-\mathcal{A})\\ \Longleftarrow\ & z_{ij}^{\text{agent}} \cdot (p_i(t|k) - p_j(t|k)) \geq S_{\mathcal{A}}(z_{ij}^{\text{agent}}) + S_{(-\mathcal{A})}(z_{ij}^{\text{agent}})\end{aligned}$$

for every $i, j \in \mathbb{N}_{[1,N_A]}$, $i \neq j$, and $t \in \{k\Delta\} \oplus \mathbb{T}_{\text{safe}}$. Figure 3 (right) illustrates the convexification for inter-agent collision avoidance.

The above convexification steps result in a semi-infinite quadratic program,

$$\underset{u_1^{\text{safe}}(k),\ldots,u_{N_A}^{\text{safe}}(k)}{\text{minimize}} \quad \sum_{i\in\mathbb{N}_{[1,N_A]}} \lambda_i \|u_i^{\text{RL}}(k) - u_i^{\text{safe}}(k)\|_2^2 \tag{10a}$$

$$\text{subject to} \quad \text{(7b)},\ \text{(7c)}$$

$$\left.\begin{matrix} i\in\mathbb{N}_{[1,N_A]},\ j\in\mathbb{N}_{[1,N_{\mathcal{K}}]}\\ t\in\{k\Delta\}\oplus\mathbb{T}_{\text{safe}}\end{matrix}\right\} \quad h_j \cdot p_i(t|k) \leq g_j - S_{\mathcal{A}}(h_j) \tag{10b}$$

$$\left.\begin{matrix} i\in\mathbb{N}_{[1,N_A]},\ j\in\mathbb{N}_{[1,N_O]}\\ t\in\{k\Delta\}\oplus\mathbb{T}_{\text{safe}}\end{matrix}\right\} \quad \begin{aligned}&z_{ij}^{\text{obs}} \cdot (p_i(t|k) - c_j)\\ &\quad \geq S_{\mathcal{O}_j}(z_{ij}^{\text{obs}}) + S_{(-\mathcal{A})}(z_{ij}^{\text{obs}}),\end{aligned} \tag{10c}$$

$$\left.\begin{matrix} i,j\in\mathbb{N}_{[1,N_A]},\ i\neq j,\\ t\in\{k\Delta\}\oplus\mathbb{T}_{\text{safe}}\end{matrix}\right\} \quad \begin{aligned}&z_{ij}^{\text{agent}} \cdot (p_i(t|k) - p_j(t|k))\\ &\quad \geq S_{\mathcal{A}}(z_{ij}^{\text{agent}}) + S_{(-\mathcal{A})}(z_{ij}^{\text{agent}}).\end{aligned} \tag{10d}$$

The optimization problem (10) has an infinite number of constraints arising from each $t \in \{k\Delta\} \oplus \mathbb{T}_{\text{safe}}$. In (10), we have the following parameters: cost scalarization weights λ_i, halfspace description $(h_j, g_j)_{j=1}^{N_{\mathcal{K}}}$ of the keep-in set $\mathcal{K}$, static obstacle information $(c_j, \mathcal{O}_j)_{j=1}^{N_O}$, rigid body of the agents $\mathcal{A}$, and user-specified separating hyperplane direction vectors z_{ij}^{obs}, z_{ij}^{agent}. Motivated by [4], we define

$$z_{ij}^{\text{obs}} = \frac{p_i(k) - c_j}{\|p_i(k) - c_j\|}, \text{ and } z_{ij}^{\text{agent}} = \frac{p_i(k) - p_j(k)}{\|p_i(k) - p_j(k)\|}. \tag{11}$$

B. Implementation

We show that (10) can be solved exactly for double integrator-based dynamics, and approximately for stabilizable linear dynamics. In both of the approaches, we formulate a quadratic program with a finite number of constraints, that can be easily solved using any off-the-shelf solver like `GUROBI` [21] or `OSQP` [22].

1) Approximate implementation for stabilizable linear dynamics: We can approximately solve the semi-infinite optimization problem (10) by sampling the constraints (10b)–(10d) at $t \in \{k\Delta\} \oplus \{\kappa\delta : \kappa \in \mathbb{N}, \kappa \leq T_{\text{safe}}^{\text{disc}}\}$ with $\delta < \Delta$ and $T_{\text{safe}}^{\text{disc}} = \lceil \frac{T_{\text{safe}}}{\delta} \rceil$. Consequently, we obtain a quadratic program with $T_{\text{safe}}^{\text{disc}}(N_A^2 + N_A N_O + N_A N_{\mathcal{K}})$ constraints.

2) Exact implementation for double integrator-based dynamics: Under the constant input (acceleration) $u_i \in \mathbb{R}^2$ over an interval $\{k\Delta\} \oplus \mathbb{T}_{\text{safe}}$, the continuous-time position of an agent with double integrator-based dynamics (3) is

$$p_i(t|k) = p_i(k) + v_i(k)(t - k\Delta) + u_i(k)((t - k\Delta)^2/2), \tag{12}$$

when starting from $p_i(k)$ with velocity $v_i(k)$ at time $k\Delta$.

For any $\ell \in \mathbb{R}^d$ and $s \in \mathbb{R}$ with $\ell \cdot p_i(k) = \ell \cdot p_i(k\Delta|k) \leq s$, consider the constraint $\ell \cdot p_i(t|k) \leq s$ for all $t \in \{k\Delta\} \oplus \mathbb{T}_{\text{safe}}$. Define $\xi : (0, T_{\text{safe}}] \to \mathbb{R}$ as $\xi(\tau; \alpha, \beta) \triangleq 2\left(\frac{\alpha}{\tau^2} - \frac{\beta}{\tau}\right)$, parameterized by $\alpha, \beta \in \mathbb{R}$ and $\alpha \geq 0$. Consequently, for $\tau = t - k\Delta$,

$$\begin{aligned}&\ell \cdot p_i(t|k) \leq s,\ \forall t \in \{k\Delta\} \oplus \mathbb{T}_{\text{safe}}\\ \iff &\ell \cdot \left(p_i(k) + v_i(k)\tau + u_i(k)(\tau^2/2)\right) \leq s,\ \forall \tau \in (0, T_{\text{safe}}]\\ \iff &\ell \cdot u_i(k) \leq \xi(\alpha, \beta),\ \forall \tau \in (0, T_{\text{safe}}] \iff \ell \cdot u_i(k) \leq \xi^*(\alpha, \beta)\end{aligned}$$

where $\alpha = s - \ell \cdot p_i(k)$, $\beta = \ell \cdot v_i(k)$, and

$$\xi^*(\alpha, \beta) \triangleq \inf_{0<\tau\leq T_{\text{safe}}} \xi(\tau; \alpha, \beta). \tag{13}$$

Note that α is non-negative by the choice of ℓ, and ξ has a non-trivial critical point $\tau_{\text{critical}} = 2\alpha/\beta$ when $\beta > 0$. The optimization problem (13) has the following solution,

$$\xi^*(\alpha, \beta) = \begin{cases} -\infty, & \alpha = 0 \text{ and } \beta > 0\\ 0, & \alpha = 0 \text{ and } \beta = 0\\ \xi(\tau_{\text{critical}}; \alpha, \beta), & \alpha, \beta > 0 \text{ and } \tau_{\text{critical}} < T_{\text{safe}}\\ \xi(T_{\text{safe}}; \alpha, \beta), & \text{otherwise.}\end{cases}$$

We summarize the exact reformulation of (10) below,

$$\text{minimize} \quad \sum\nolimits_{i\in\mathbb{N}_{[1,N_A]}} \lambda_i \|u_i^{\text{RL}}(k) - u_i^{\text{safe}}(k)\|_2^2 \tag{14a}$$

$$\text{subject to} \quad \text{(7b)},\ \text{(7c)}$$

$$\left.\begin{matrix} i\in\mathbb{N}_{[1,N_A]}\\ j\in\mathbb{N}_{[1,N_{\mathcal{K}}]}\end{matrix}\right\} \quad h_j \cdot u_i^{\text{safe}}(k) \leq \xi^*(\alpha_{ij}^{\text{keep}}, \beta_{ij}^{\text{keep}}), \tag{14b}$$

$$\left.\begin{matrix} i\in\mathbb{N}_{[1,N_A]}\\ j\in\mathbb{N}_{[1,N_O]}\end{matrix}\right\} \quad -z_{ij}^{\text{obs}} \cdot u_i^{\text{safe}}(k) \leq \xi^*(\alpha_{ij}^{\text{obs}}, \beta_{ij}^{\text{obs}}), \tag{14c}$$

$$\left.\begin{matrix} i,j\in\mathbb{N}_{[1,N_A]}\\ i\neq j\end{matrix}\right\} \quad z_{ij}^{\text{agent}} \cdot (u_j^{\text{safe}}(k) - u_i^{\text{safe}}(k)) \leq \xi^*(\alpha_{ij}^{\text{agent}}, \beta_{ij}^{\text{agent}}). \tag{14d}$$

with $\alpha_{ij}^{\text{keep}} = g_j - S_{\mathcal{A}}(h_j) - h_j \cdot p_i(k)$, $\beta_{ij}^{\text{keep}} = h_j \cdot v_i(k)$, $\alpha_{ij}^{\text{obs}} = z_{ij}^{\text{obs}} \cdot (p_i(k) - c_j) - S_{\mathcal{O}}(z_{ij}^{\text{obs}}) - S_{(-\mathcal{A})}(z_{ij}^{\text{obs}})$, $\beta_{ij}^{\text{obs}} = -z_{ij}^{\text{obs}} \cdot v_i(k)$, $\alpha_{ij}^{\text{agent}} = z_{ij}^{\text{agent}} \cdot (p_i(k) - p_j(k)) - S_{\mathcal{A}}(z_{ij}^{\text{agent}}) - S_{(-\mathcal{A})}(z_{ij}^{\text{agent}})$, and $\beta_{ij}^{\text{agent}} = z_{ij}^{\text{agent}} \cdot (v_j(k) - v_i(k))$. In contrast to the approximate quadratic program (Section III-B.1), (14) has $(N_A^2 + N_A N_O + N_A N_{\mathcal{K}})$ constraints, a reduction by factor of $T_{\text{safe}}^{\text{disc}}$.

C. Safety properties

We state the safety properties of the proposed approach. We show that (10) provides continuous-time safety. Furthermore, the sampling-based approximation in Section III-B.1 addresses Problem 1, and (14) addresses Problem 2.

Proposition 1 (General safety property)**.** *We assume that $T_{safe} > \Delta$ is sufficiently large such that* (10) *is feasible at every discrete time step $k \in \mathbb{N}$. Then, the application of safety filter* (10) *at every k will keep the system collectively safe for all $t \geq 0$.*

The proof of Proposition 1 follows by induction. By construction, the feasibility of (10) at every discrete-time step k is sufficient to guarantee collective safety at all times. The feasibility assumption is necessary to avoid pathological cases where the safety filter, the choice of the horizon T_{safe}, and the control constraints precludes a safe behavior. For example, an initial configuration where an agent is just beside an obstacle and the initial velocity of the agent is towards the obstacle can not be rendered safe by the safety filter.

1) Guarantees for the approximate approach: For the approximate implementation, the collective safety guarantee offered by Proposition 1 only holds at every discrete time $k\delta$, $k \in \mathbb{N}$. Such a sampling-based quadratic program addresses Problem 1.

In our numerical simulations, the relatively strong assumption of the feasibility of (10) at all discrete time steps $k \in \mathbb{N}$ in Proposition 1 was never violated for T_{safe} sufficiently larger than Δ ($T_{\text{safe}} = 2$ s, $\Delta = 0.1$ s, and $\delta = 0.01$ s). Instead of uniform sampling, one can also obtain probabilistic guarantees of safety via a scenario approach [23].

2) Guarantees for the exact approach: In case of the double integrators, we show that (14) does not suffer from trivial infeasibility due to $\xi^* = -\infty$.

Proposition 2 (Ruling out trivial infeasibility of (14))**.** *Assume that* (14) *is feasible at the initial time step $k = 0$, and the sets $\mathcal{O}_j$ and $\mathcal{A}$ are balls of radius $r_j^{obs} > 0$, $j \in \mathbb{N}_{[1,N_O]}$ and $r^{agent} > 0$ respectively. Then, for a safety horizon $T_{safe} > \Delta$, the constraints* (14b)–(14d) *defined at any time $k\Delta$ individually defines a non-empty feasible region (or is not trivially infeasible) for all $t \in \{k\Delta\} \oplus \mathbb{T}_{safe}$.*

Proof. Proof by induction. The base case ($k = 0$) holds.

(14b) *never occurs with $\alpha = 0, \beta > 0$:* Assume for induction that the constraints (14b) is not trivially empty for some time step $k \in \mathbb{N}$ for some $i \in \mathbb{N}_{[1,N_A]}$ and $j \in \mathbb{N}_{[1,N_{\mathcal{K}}]}$. Define $\alpha(t) = g_j - h_j \cdot p_i(t|k)$ and $\beta(t) = h_j \cdot p_i(t|k)$. We know $\alpha(t) \geq 0$ for every $t \in \{k\Delta\} \oplus \mathbb{T}_{\text{safe}}$, since $T_{\text{safe}} > \Delta$. Consequently, we have $\nabla_t \alpha(t) \geq 0$ whenever $\alpha(t) = 0$ for some $t \in \{k\Delta\} \oplus \mathbb{T}_{\text{safe}}$. We observe that $\nabla_t \alpha(t) = -h_j \cdot v_i(t|k) = -\beta(t)$. Thus, at discrete time $k+1$, we have $\beta \leq 0$ whenever $\alpha = 0$, as desired.

(14c) *never occurs with $\alpha = 0, \beta > 0$:* Assume for induction that the constraint (14c) is not trivially empty for some time step $k \in \mathbb{N}$ for some $i \in \mathbb{N}_{[1,N_A]}$ and $j \in \mathbb{N}_{[1,N_O]}$. Define $\alpha_k(t) = \|p_i(t|k) - c_j\| - (r_j^{\text{obs}} + r^{\text{agent}})$. Consequently, we have $\beta(t) = -\nabla_t \alpha_k(t) = 2(p_i(t|k) - c_j) \cdot v_i(t|k)$. By same calculus arguments as before, we have $\beta \leq 0$, when $\alpha = 0$ ($\alpha_k((k+1)\Delta)$).

The proof of (14d) follows similarly to that of (14c). $\square$

Since (14) is an exact reformulation of (10), (14) addresses Problem 2 by Proposition 1. Proposition 2 does not eliminate the possibility of infeasibility arising from the intersection of the constraints in (14). A possible approach to guarantee recursive feasibility of (14) is by leveraging existing results in model predictive controls literature [24], which we will explore in future work.

IV. Experiments

We validate the proposed approach in simulation and check the feasibility of the approach in hardware experiments. We show that our proposed approach is real-time implementable, reduces the training effort for obtaining a safe RL-based motion planner without compromising on the generalizability, and scales well with the number of agents. All simulations and training were done on an Ubuntu workstation with Intel Xeon 3.3 GHz 12-core CPU, 192 GB memory, and two Nvidia RTX 2080 Ti GPUs.

A. Safe multi-agent motion planning: Software and hardware experiments

Our experimental setup is based on the Crazyflie 2.1 quadrotors. We flew six Crazyflies in a 3m$\times$3m$\times$1.5m volume using OptiTrack motion-capture system (see Figure 1). We obtained the marker point cloud at 120Hz, and used the Crazyswarm package to control the Crazyflies. We used the onboard PID controllers of the Crazyflies to track the waypoints (2D position with a constant altitude) sent at 10Hz.

We chose $\Delta = 0.1$ s, $\delta = 0.01$ s, $T_{\text{safe}} = 2$ s, and $r_A = 0.15$ m. We designed the single-agent RL-based motion planner using the proximal policy optimization (PPO) algorithm from `stable-baselines` [19]. We trained the PPO algorithm for a total of 10^7 time steps, with default parameters of `stable-baselines` with the entropy coefficient modified to 0.001. The data generation and the policy training for the agent took around 9 hours. We used `GUROBI` [21] to solve the quadratic program associated with the safety filter.

Figure 4 shows a typical run of one of the experiments. We studied the utility of the proposed solution (RL-based control and safety filtering) by comparing it with using the safety filter with a proportional controller, i.e., $a_i(k) = 0$. The RL-based controller took significantly less time to complete the task (navigate the agents safely towards their targets) than the baseline controller (27.1 s instead of 60 s). The RL-based controller leverages past experience to encourage motion around the obstacle, as compared to the proportional controller.

Figure 5 shows that the overall computation time per iteration, including the overhead of evaluating the neural policy and the computation of the safe actions using CVXPY [25] and GUROBI [21]. We see that the overall computation times per iteration in both of the approaches are below 0.1 s, which enable a real-time implementation at 10Hz. The compute times may be further reduced by shifting from a Python implementation to `C` or `C++`.

Figure 6 shows the performance of the proposed algorithm over a randomly chosen set of initial configurations. None of

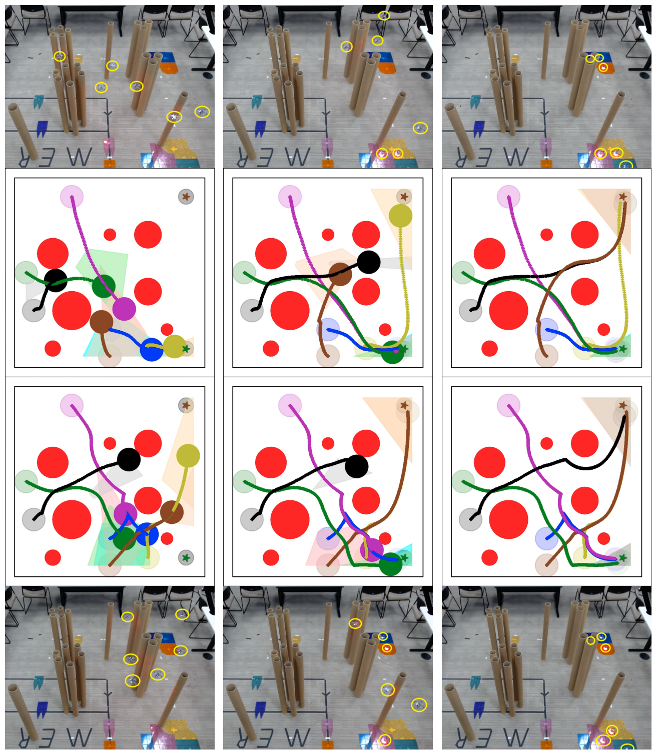


Fig. 4. Safe multi-agent motion planning using the proposed safety filter in conjunction with RL-based controllers and a classical proportional controller. (Top two rows) Snapshots of the hardware experiments and illustrations of the agent trajectories when using the RL-based motion planner at times $t \in \{90, 180, 271\}$. (Bottom two rows) Trajectories of the agents when using the baseline motion planner at times $t \in \{200, 400, 600\}$, along with the snapshots of the hardware experiments. Red circles denote the obstacles, shaded regions indicate static obstacle-free positions at each iteration (determined via convexification), shaded circles denote the agent starting points, and starred circles denote the targets.

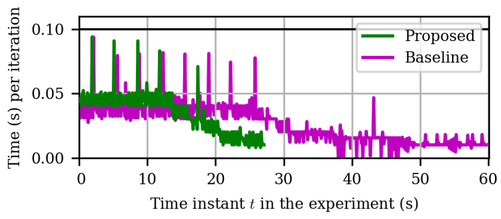


Fig. 5. Overall computation time per iteration is always below 0.1 seconds.

these configurations led to an infeasible quadratic program for the safety filter or a violation of the collective safety. In more than 80% of the cases, the proposed approach completed the task. In the rest of the cases, it still ensured safety but resulted in a deadlock. Several strategies exist to resolve deadlocks, including the use of KKT conditions [26], which we will explore in future work.

B. Safety filter reduces the training effort for safe RL

Figure 7 shows the improvement in the quality of the single-agent motion planner with the application of the safety filter. The use of safety filter with RL-based motion planner ensures that the agent reaches the target from almost all feasible starting positions. For a small fraction ($< 2\%$) of the feasible state space, the safety filter causes the agent to

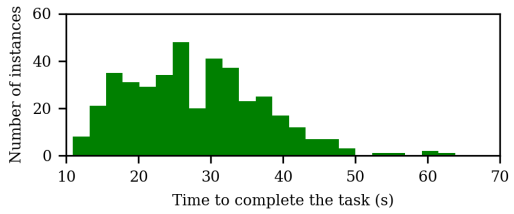


Fig. 6. Time taken by the proposed approach to complete the task from 500 random initial configurations for the agents. 80.6% of the configurations successfully completed the task, while the rest resulted in deadlock.

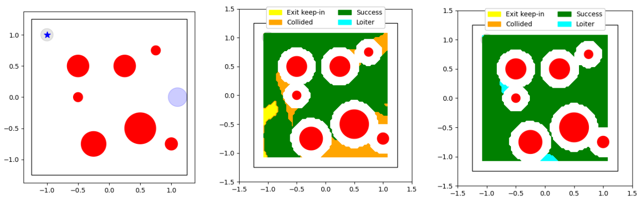


Fig. 7. Safety filter significantly improves the safety and performance of the RL-based motion planner. (Left) Original environment for the single-agent motion planner. (Middle) Evaluation of the standalone RL-based motion planner when starting from different feasible positions. (Right) Evaluation of the proposed approach (RL-based motion planner with safety filter).

"loiter", i.e., the filter ensures collective safety at the expense of performance (reaching the target). Without the safety filter, the RL-based motion planner suffers from collision with the static obstacle or exits the safe region when starting in a significant region of the state space.

C. Safety filter supports real-time implementation with a moderately high number of agents

To perform scalability analysis of the safety filter, we reduced r_A to 0.05, and collected computational times for the safety filter from three separate trials with randomly initialized locations for the agents in simulation. Figure 8 shows that the compute time of the safety filter increases only moderately with the number of agents. The scalability of the approach can be attributed to the simplicity of the convex quadratic program solved by the safety filter.

V. Conclusion

This paper proposed a novel optimization-based safety filter that guarantees collective safety of a multi-agent system driven by a learning-based motion planner designed for a single agent. We formulate the safety filter as a quadratic program that can be easily solved using off-the-shelf solvers, enabling a real-time implementable approach to safe multi-agent motion planning.

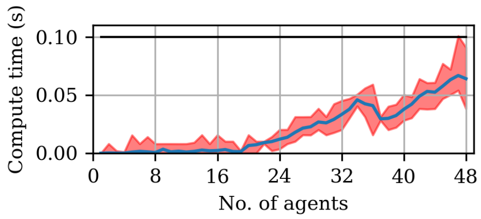


Fig. 8. Computation time of the safety filter increases moderately with the number of agents

REFERENCES

[1] M. Radmanesh and M. Kumar, "Flight formation of UAVs in presence of moving obstacles using fast-dynamic mixed integer linear programming," *Aerospace Science and Technology*, vol. 50, pp. 149–160, 2016.

[2] A. Sud, R. Gayle, E. Andersen, S. Guy, M. Lin, and D. Manocha, "Real-time navigation of independent agents using adaptive roadmaps," in *ACM SIGGRAPH 2008 classes*, 2008, pp. 1–10.

[3] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[4] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *2012 IEEE/RSJ international conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1917–1922.

[5] M. Srinivasan, S. Coogan, and M. Egerstedt, "Control of multi-agent systems with finite time control barrier certificates and temporal logic," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 1991–1996.

[6] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, vol. 39, no. 1, pp. 101–121, 2015.

[7] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li, and L. Wang, "Joint optimization of multi-UAV target assignment and path planning based on multi-agent reinforcement learning," *IEEE access*, vol. 7, pp. 146 264–146 272, 2019.

[8] L. Lv, S. Zhang, D. Ding, and Y. Wang, "Path planning via an improved DQN-based learning policy," *IEEE Access*, vol. 7, pp. 67 319–67 330, 2019.

[9] S. H. Semnani, H. Liu, M. Everett, A. de Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3221–3226, 2020.

[10] M. Srinivasan, A. Chakrabarty, R. Quirynen, N. Yoshikawa, T. Mariyama, and S. Di Cairano, "Fast multi-robot motion planning via imitation learning of mixed-integer programs," in *Proceedings of IFAC Modeling, Estimation and Control Conference (MECC)*, 2021.

[11] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6382–6393.

[12] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

[13] R. Cheng, M. J. Khojasteh, A. D. Ames, and J. W. Burdick, "Safe multi-agent interaction through robust control barrier functions with learned uncertainties," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 777–783.

[14] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Transactions on Automatic Control*, 2021.

[15] I. ElSayed-Aly, S. Bharadwaj, C. Amato, R. Ehlers, U. Topcu, and L. Feng, "Safe multi-agent reinforcement learning via shielding," *arXiv preprint arXiv:2101.11196*, 2021.

[16] I. Kolmanovsky and E. G. Gilbert, "Theory and computation of disturbance invariant sets for discrete-time linear systems," *Mathematical problems in engineering*, vol. 4, no. 4, pp. 317–367, 1998.

[17] I. Kolmanovsky, E. Garone, and S. Di Cairano, "Reference and command governors: A tutorial on their theory and automotive applications," in *2014 American Control Conference*, 2014, pp. 226–241.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[19] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," *GitHub repository*, 2019.

[20] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[21] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com

[22] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[23] M. C. Campi, S. Garatti, and M. Prandini, "The scenario approach for systems and control design," *Annual Reviews in Control*, vol. 33, no. 2, pp. 149–157, 2009.

[24] J. Löfberg, "Oops! I cannot do it again: Testing for recursive feasibility in MPC," *Automatica*, vol. 48, no. 3, pp. 550–555, 2012.

[25] S. Diamond and S. Boyd, "CVXPY: A python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.

[26] J. Grover, C. Liu, and K. Sycara, "Deadlock analysis and resolution for multi-robot systems," *Workshop on Algorithmic Foundations of Robotics*, 2021.