LiveNet: Robust, Minimally Invasive Multi-Robot Control for Safe and Live Navigation in Constrained Environments

A Research Paper submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Srikar Gouru

Spring, 2025

Technical Project Team Members

Srikar Gouru

Siddarth Lakkoju

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Rohan Chandra, Department of Computer Science

Proceedings of Machine Learning Research vol 283:1–13, 2025

LIVENET: Robust, Minimally Invasive Multi-Robot Control for Safe and Live Navigation in Constrained Environments

Srikar Gouru MCJ2VB@VIRGINIA.EDU
University of Virginia

Siddharth Lakkoju QMN4CJ@VIRGINIA.EDU
University of Virginia

Rohan Chandra ROHANCHANDRA@VIRGINIA.EDU
University of Virginia

Abstract

Robots in densely populated real-world environments frequently encounter constrained and cluttered situations such as passing through narrow doorways, hallways, and corridor intersections, where conflicts over limited space result in collisions or deadlocks among the robots. Current decentralized state-of-the-art optimization- and neural network-based approaches (i) are predominantly designed for general open spaces, and (ii) are overly conservative, either guaranteeing safety, or liveness, but not both. While some solutions rely on centralized conflict resolution, their highly invasive trajectories make them impractical for real-world deployment. This paper introduces LIVENET, a fully decentralized and robust neural network controller that enables human-like yielding and passing, resulting in agile, non-conservative, deadlock-free, and safe, navigation in congested, conflict-prone spaces. LIVENET is minimally invasive, without requiring inter-agent communication or cooperative behavior. The key insight behind LIVENET is a unified CBF formulation for *simultaneous* safety and liveness, which we integrate within a neural network for robustness. We evaluated LIVENET in simulation and found that general multi-robot optimization- and learning-based navigation methods fail to even reach the goal, and while methods designed specially for such environments do succeed, they are 10–20$\times$ slower, 4–5$\times$ more invasive, and much less robust to variations in the scenario configuration such as changes in the start states and goal states, among others. We open-source the LIVENET code at **https://github.com/srikarg89/LiveNet**.

Keywords: Multi-Robot Navigation, Liveness, Safety, Constrained Environments.

1. Introduction

Large-scale multi-agent robot navigation has recently gained popularity for its applications in many fields, including warehouse robots, autonomous vehicles, unmanned aerial vehicles, and more (Bogue (2024); Rasheed et al. (2022)). These systems frequently operate in constrained and cluttered environments, such as navigating doorways, intersections, or narrow hallways. Such scenarios often lead to conflicts, manifesting as deadlocks, collisions, or both, when multiple agents attempt to occupy the same limited space simultaneously (Chandra et al. (2024)). In contrast, humans navigate these challenges effortlessly and intuitively, demonstrating agility and safety by dynamically modulating their speed and trajectory. This allows them to avoid collisions (ensuring safety) and prevent deadlocks (maintaining liveness—defined as the ability to continually make progress toward their goal) while ensuring smooth and efficient transitions. This work investigates a fundamental research question: *How can robots emulate human-like agility, safety, and liveness in constrained environments?*

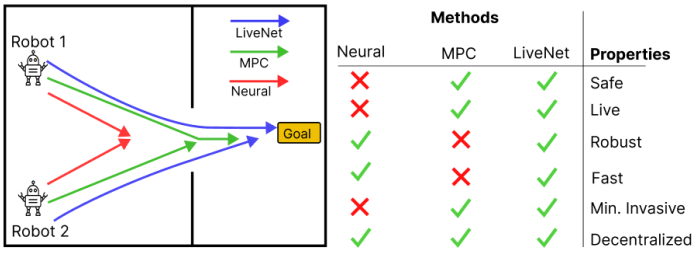


Figure 1: LIVENET enables minimally invasive, robust, safe and deadlock-free navigation in constrained environments compared to existing methods.

We approach this question by designing a low-level controller capable of *simultaneously* ensuring safety and liveness. Achieving only one of these properties is insufficient for replicating human-like behavior, such as agile yielding. Focusing solely on safety results in overly conservative behavior, while prioritizing liveness alone can lead to aggressive and potentially unsafe navigation. While some efforts have attempted to integrate both safety and liveness (Chandra et al. (2024); Wang et al. (2017); Zhou et al. (2017); Chen et al. (2023); Garg et al. (2024)), these solutions often perturb the robots in an invasive manner that forces them to adopt suboptimal trajectories. For example, (Zhou et al. (2017); Wang et al. (2017)) implement the right-hand-rule to induce clockwise movement in the event of a deadlock. Furthermore, in prioritizing safety, current methods only resolve deadlocks after they occur and all agents come to a stop. This is in stark contrast to how humans navigate by preemptively detecting and preventing the deadlock, without invasive maneuvers or delays. These issues highlight a critical gap in the literature towards achieving a robust, minimally invasive liveness without sacrificing safety. An optimal algorithm to navigating in constrained and cluttered spaces, therefore, must satisfy the following criteria:

1. *Decentralized / Non-cooperative*: The algorithm should not assume that agents can communicate with one another or with a centralized controller. The robots should only gain information about their surrounding environment through sensory perception.
2. *Robust*: Agents must adapt to changes in the environment or in their own configurations.
3. *Safe*: The agents should maintain a predefined safety distance from other agents and obstacles.
4. *Live*: The agents should avoid deadlocks and continuously make progress towards the goal.
5. *Minimally invasive*: An optimal strategy should apply the least disruptive intervention using minimal perturbations to ensure conflict resolution.
6. *Dynamically feasible*: The agents should follow predefined non-holonomic and kinodynamic constraints, including bounds on the velocity and acceleration of the agent.

Main Contributions: We propose a novel approach for optimal end-to-end learnable multi-robot navigation in constrained spaces such as doorways and corridor intersections. The key insight of our approach is to formulate both safety as well as liveness via differentiable CBFs within the neural network controller.

- We propose the first safe, robust neural controller with provable liveness guarantees (Theorem (1)) for agile and smooth multi-robot navigation in constrained spaces.
- Unlike prior methods, LIVENET, while still fully decentralized, is minimally invasive, only perturbing the speed of a robot by the smallest amount necessary, without changing its direction, resulting in smoother and more optimal trajectories.
- LIVENET's control cycle frequency is between 10–20$\times$ faster than MPC-based optimization approaches and 20$\times$ faster than MACBF (Qin et al. (2021)), a state of the art end-to-end learning-based multi-robot navigation baseline.
- LIVENET is robust to changes in the environment and agent configurations. Given a wide range of diverse environments, LIVENET succeeded in 30% more scenarios compared to MPC-based baselines.

2. Related Work

In this section we discuss the variety of methods that have been applied to safely navigate multi-robot deadlock avoidance scenarios.

2.1. Simultaneous Safety and Deadlock Resolution Methods

Most current methods rely on cooperative, predetermined behavior between the robots to resolve conflicts. For instance, in (Zhou et al. (2017); Wang et al. (2017); Zhu et al. (2022); Chen et al. (2023)), the authors heuristically define a clockwise movement to establish the right of way (the rightmost agent moves first). Other deadlock resolution methods generate vehicle priorities through reservation systems like first come first serve (Au et al. (2015)) or auctions with predefined bidding

strategies (Carlino et al. (2013); Suriyarachchi et al. (2022)). Garg et al. (Garg et al. (2024)) used large language models to act as a central arbiter to resolve the conflict. However, these heuristics result in larger perturbations than necessary to avoid the deadlock, and they often falter in scenarios with unpredictable agents where knowledge of other agents' planned trajectories is unavailable.

2.2. Other Multi-Robot Navigation Approaches

Multi-agent path-finding (MAPF) algorithms such as conflict-based search (CBS) and its variants (Sharon et al. (2015)), M^* (Stern (2019)), ICTS (Sharon et al. (2013)), and Uniform Cost Search (Mao et al. (2024), Guo et al. (2024), McNaughton et al. (2011)). Many MAPF techniques yield a globally optimal solution, but require centralized solvers, discretized state spaces, and low-dimensional observation, state, and action spaces, thereby restricting its use primarily to simulation or offline as a coarse preprocessing step (Ma (2022)). These assumptions are prohibitive to most real world robots that possess non-holonomic and kinodynamic constraints. Learning-based approaches use imitation learning (Hussein et al. (2018); Yan et al. (2022); Daftry et al. (2017); Qin et al. (2021); Xiao et al. (2023)) and multi-agent reinforcement learning (MARL) (Liu et al. (2020); Martinez-Gil et al. (2012); Mehr et al. (2023)) to learn a navigation policy using supervised or unsupervised learning methods. While these methods offer robustness and scalability, they model safety as a learned behavior rather than a constraint. Thus, despite large training sets, no mathematical guarantees for robot safety can be drawn, which is crucial for robots deployed in safety-critical environments.

Optimization-based methods, particularly Model Predictive Control (MPC) with control barrier functions (CBFs), have been employed to calculate safe trajectories over short future horizons (Mestres et al. (2024), Zhu et al. (2020)). These receding-horizon control strategies iteratively solve an optimization problem at each step, adjusting the agent's trajectory to avoid collisions in the immediate future. Chandra et al. (Chandra et al. (2024)) model liveness as a control barrier function (CBF) within a receding-horizon control scheme, adjusting the agent's trajectory to avoid collisions and deadlocks in the immediate future. Although effective in certain settings, these methods lack robustness since they are often tuned to specific environment configurations and quickly falter when the environment changes. Another class of distributed optimization-based methods uses dynamic game theory to compute a Nash equilibria for similar problems that dictates all agents' trajectories (Wang et al. (2021a,b); Schwarting et al. (2021); Sun et al. (2015, 2016); Morimoto and Atkeson (2003); Fridovich-Keil et al. (2020); Di and Lamperski (2018)). However, this requires knowledge of the other agents' objective functions, their desired trajectories, and their kinodynamic constraints (Mylvaganam et al. (2017)).

Symbol	Description
Problem formulation (Section 3)	
k	Number of agents
T	Planning horizon
$\mathcal{X}$	General continuous state space
$\mathcal{X}_I$	Set of initial states
$\mathcal{X}_g$	Set of final states
x_t^i	State of agent i at time t
Ω^i	Observation set of agent i
$\mathcal{O}^i : \mathcal{X} \to \Omega^i$	Agent i's observation function
o_t^i	Observation of agent i at time t
Γ^i	Agent i's trajectory
$\widetilde{\Gamma}^i$	Agent i's preferred or desired trajectory
Ψ^i	Agent i's input control sequence
$\mathcal{T} : \mathcal{X} \times \mathcal{U}^i \to \mathcal{X}$	Environment transition dynamics (Equation 1b)
$\mathcal{U}^i$	Action space for agent i
$\mathcal{J}^i$	Running cost for agent i ($\mathcal{J}_t^i : \mathcal{X} \times \mathcal{U}^i \to \mathbb{R}$)
$\mathcal{J}_T^i$	Terminal cost at time T
$\mathcal{C}^i\left(x_t^i\right) \subseteq \mathcal{X}$	Convex hull of agent i
$\overline{\Gamma}^i$	Agent i's minimally invasive trajectory
$z_t^i \in \mathcal{X} \times \Omega^i$	Model input, consisting of x_t^i and o_t^i.
$\mathcal{F} : \mathcal{X} \times \Omega^i \to \mathcal{U}^i$	network defining agent i's controller
Technical Approach (Section 4)	
$b^i\left(z_t^i\right) : \mathcal{X} \times \Omega^i \longrightarrow \mathbb{R}$	Control barrier function (CBFs)
$b_o^i, b_l^i\left(z_t^i\right)$	Obstacle and liveness CBFs
$L_f b^i\left(z_t^i\right), L_g b^i\left(z_t^i\right)$	Lie derivatives of $b^i\left(z_t^i\right)$ w.r.t f and g.
s_t^i, θ_t^i, v_t^i	Position, heading, and velocity of agent i
$\widetilde{\Gamma}^i, \widetilde{\Psi}^i$	State and input controls trajectory dataset
p^o, p^l	Penalty values defining the relaxation of the CBFs

Table 1: Summary of notation used in this paper.

The algorithms described above are able to either guarantee safety, liveness, or robustness to adapt to new scenarios well, but are unable to do all three. This research builds on foundational ideas of Neural Network based controllers and Control Barrier Functions (CBFs) (Wang et al. (2017)) to provide safety and liveness while being robust.

3. Problem Formulation

In this section, we formulate the problem objective that we aim to solve. Notation for variables referenced is summarized in table 1. We formulate the problem as the following partially observable stochastic game (POSG) (Hansen et al. (2004)): $\langle k, T, \mathcal{X}, \mathcal{U}^i, \mathcal{T}, \mathcal{J}^i, \mathcal{O}^i, \Omega^i \rangle$, where k refers to the number of agents, and T refers to the finite horizon length of the game. A superscript of i refers to the i^{th} agent, where $i \in [1, .., k]$ and a subscript of t refers to discrete time step t where $t \in [1, .., T]$. At any given time step t, agent i has state $\mathbf{x}_t^i \in \mathcal{X}$ where $\mathcal{X}$ is the general, continuous state space. $\mathcal{U}^i$ is the continuous control space for robot i representing the set of admissible inputs for i. Agent dynamics are defined by the transition function $\mathcal{T} : \mathcal{X} \times \mathcal{U}^i \to \mathcal{X}$ at time $t \in [1, .., T-1]$. The cost function, $\mathcal{J}^i : \mathcal{X} \times \mathcal{U}^i \to \mathbb{R}$ is used to determine the cost of the specified control action in the agent's current state, and the terminal cost $\mathcal{J}_T^i : \mathcal{X} \to \mathbb{R}$ is used to calculate the cost of the terminal state $\mathbf{x}_T^i$. Each agent i also has an observation $o_t^i \in \Omega^i$ which is determined via the observation function $o_t^i = \mathcal{O}^i(x_t^i)$. A discrete trajectory of agent i is defined by $\Gamma^i = (\mathbf{x}_0^i, \mathbf{x}_1^i, ..., \mathbf{x}_T^i)$, and has a corresponding control input sequence $\Psi^i = (u_0^i, ..., u_{T-1}^i)$. Agents follow the control-affine dynamics $\mathbf{x}_{t+1}^i = f(\mathbf{x}_t^i) + g(\mathbf{x}_t^i)u_t^i$, where f, g are locally Lipschitz continuous functions. At any time t, each agent i occupies a space given by $C^i(\mathbf{x}_t^i) \subseteq \mathcal{X}$. Two robots i, j are considered colliding at time t if $C^i(\mathbf{x}_t^i) \cap C^j(\mathbf{x}_t^j) \neq \emptyset$.

A *Social Mini-Game* (SMG) is a variation of the generic POSG where each agent has a starting state, $\mathbf{x}_0^i \in \mathcal{X}_I$ and a goal state $\mathbf{x}_g^i \in \mathcal{X}_g$ where $\mathcal{X}_I$ and $\mathcal{X}_g$ are subsets of the continuous space $\mathcal{X}$ (Chandra et al. (2024)). Additionally, each agent has a preferred trajectory, denoted by $\widetilde{\Gamma}^i$, which would be the desired trajectory that the agent would take in the absence of any other agents:

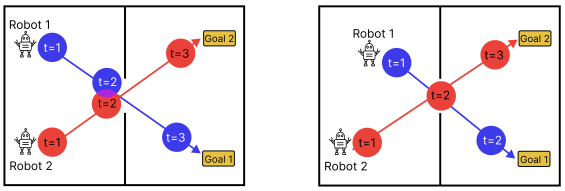
$$\left(\widetilde{\Gamma}^i, \widetilde{\Psi}^i\right) = \arg\min_{(\Gamma^i, \Psi^i)} \sum_{t=0}^{T-1} \mathcal{J}^i\left(\mathbf{x}_t^i, u_t^i\right) + \mathcal{J}_T^i\left(\mathbf{x}_T^i\right) \tag{1a}$$

$$\text{s.t } \mathbf{x}_{t+1}^i = f\left(\mathbf{x}_t^i\right) + g\left(\mathbf{x}_t^i\right) u_t^i, \quad \forall t \in [1; T-1] \tag{1b}$$

$$u_{min} \leq u_t^i \leq u_{max} \tag{1c}$$

$$\mathbf{x}_0^i \in \mathcal{X}_I, \quad \mathbf{x}_T^i \in \mathcal{X}_g \tag{1d}$$

A game is considered a social mini-game when if for some $t \in [1, .., T]$, there exists at least one pair i, j where $i \neq j$ such that $C^i(\mathbf{x}_t^i) \cap C^j(\mathbf{x}_t^j) \neq \emptyset$ where $\mathbf{x}_t^i \in \widetilde{\Gamma}^i$ and $\mathbf{x}_t^j \in \widetilde{\Gamma}^j$. As shown in Figure 2, if the desired spatio-temporal trajectories of any two agents result in either a collision or a deadlock, the game is considered an SMG. As in previous works (Chandra et al. (2024), Grover et al. (2023)), we define a deadlock when a robot i is in deadlock if the current velocity $v_t^i = 0$ prior to reaching the goal, meaning $\mathbf{x}_t^i \notin \mathcal{X}_g$. A robot exhibits *liveness* if it is able to prevent collisions and deadlocks.



(*a*) SMG Scenario (*b*) Non-SMG Scenario

Figure 2: Example SMG and Non-SMG scenarios with agent 1's desired trajectory in red and agent 2's desired trajectory in blue. Their starting and goal locations are indicated by $t = 1$ and $t = 4$, respectively, with t being used to show the agents' time-parameterized desired trajectories. Collisions are shown in purple.

Objective: Our goal is to prevent both collisions and deadlocks in an SMG by perturbing the preferred trajectory, $\widetilde{\Gamma}^i$, in a *minimally invasive* manner. We define the new trajectory, $\overline{\Gamma}^i$ to be minimally invasive if it perturbs the agent's velocity throughout the trajectory by the minimal amount possible without any spatial deviation from the preferred trajectory and avoids collisions with all other agents while following kinodynamic constraints. This mimics human yielding when crossing an intersection or passing through a doorway, where humans simply slow down to allow someone else to pass, but don't change their intended spatial path. For a minimally invasive trajectory, we define the cost function $\mathcal{J}^i$ as the time to reach the

goal. This implies wanting to travel at the maximum allowed velocity for as long as possible, thus we want to minimize any velocity deviations. Mathematically, this means that over the planning time-horizon T, the following constraints must be incorporated into Problem (1):

$$\sum_{t=0}^{T-1} D(\overline{s}_t^i, \widetilde{\Gamma}) \leq \epsilon^{(1)}, \qquad \sum_{t=1}^{T-1} |\overline{v}_t^i - \overline{v}_{t-1}^i| \leq \epsilon^{(2)}, \qquad C^i(\overline{\mathbf{x}}_t^i) \cap C^j(\mathbf{x}_t^j) = \emptyset, \qquad \overline{v}_t^i = 0 \leftrightarrow \overline{\mathbf{x}}_t^i \notin X_g \quad (2)$$

for all $i, j \in [0, k]$ s.t. $i \neq j$ where $D(\overline{s}_t^i, \widetilde{\Gamma})$ represents the spatial deviation of point $\overline{s}^i$ from the desired trajectory $\widetilde{\Gamma}$. Formally, our goal is now to solve Problem (1) with the added constraints given by (2). We define a minimally invasive solution as one that minimizes the value of $\epsilon^{(1)}$, and in the event of a tie minimizes $\epsilon^{(2)}$.

4. LIVENET: Technical Approach

Each LIVENET agent is defined by a neural network, where the function $\mathcal{F} : \mathcal{X} \times \Omega^i \rightarrow \mathcal{U}^i$ defines the feed-forward function of the model. The input to the network is the agent's state and observation, namely $z_t^i = (\mathbf{x}_t^i, o_t^i)$. The cost function for the model is defined via a mean-squared-error loss function such that $\mathcal{J}^i = \frac{1}{T}\sum_{t=0}^{T}(F(\widehat{z}_t^i) - \widehat{u}_t^i)^2$ for all corresponding $(\widehat{z}^i, \widehat{u}_t^i)$ in a trajectory dataset $(\widehat{\Gamma}^i, \widehat{\Psi}^i)$. The LIVENET network (visualized in Figure 3) takes in the agent's state and observation as inputs, and passes it through three subnetworks, F_r, F_o, and F_l, which generate the reference control output u_{ref}, along with penalty values for the obstacle barrier function, p^o and the liveness barrier function, p^l. p^o determines the level of relaxation on the obstacle CBF (Section 4.1) and p^l determines the level of relaxation on the liveness CBF (Section 4.2). The network subsequently feeds these variables into a quadratic programming (QP) layer (Amos and Kolter (2017)) to minimize the function $(u - u_{ref})^2$ given differential CBF (dCBF) constraints defined by p^o and p^l. During backpropagation, the mean squared error loss between the outputted u and the optimal $\hat{u}$ (provided via the training dataset) is used to optimize the weights in the networks F_r, F_o, and F_l via gradient descent.

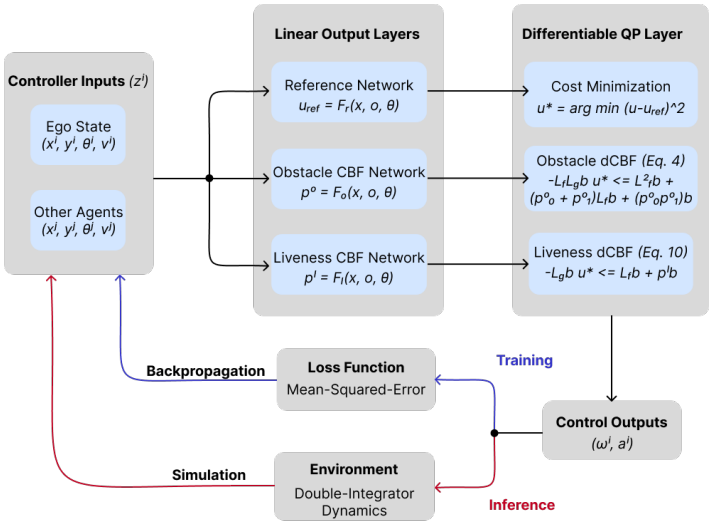


Figure 3: **LIVENET Architecture Overview** The ego state and observation inputs get fed into a feedforward network with three individual outputs: the reference control, the obstacle penalties, and the minimal invasiveness penalties. These three outputs get fed into a differentiable QP layer which solves a standard QP problem with inequality constraints (Equations (4) and (11)) to enforce the CBFs. During backpropagation, the optimal reference value, as well as optimal penalty values for the CBF constraints, are learned.

The OptNet framework (Amos and Kolter (2017)) that LIVENET is built on allows QP problems to be passed in with generalized inequality constraints in the form $G(z)u \leq h(z)$. Sections 4.1 and 4.2 discuss the usage of Higher-Order CBFs (HOCBFs) in order to construct barrier function inequalities that are dependent on the control inputs. We refer the reader to [Xiao and Belta (2019)] for more details. Our approach presents both a multi-agent differential CBF (dCBF) layer

for multi-dimensional state spaces, as well as a liveness dCBF filter to ensure deadlock avoidance in a minimally invasive manner. In this work, we specifically explore the $k = 2$ scenario.

4.1. Multi-Agent Collision Avoidance dCBFs

We present an environment setup with double-integrator unicycle dynamics and a state space defined by $\mathbf{x} = (x, y, \theta, v) \in \mathcal{X}$ where x and y represent the 2D position of the robot, θ represents the heading, and v represents the forward velocity. When referring to the robot's position, $s = (x, y)$ is used. The control inputs are defined by $(\omega, a) \in \mathcal{U}^i$ where ω represents turning velocity and a represents linear acceleration. Observations of agent i includes the positions, headings, and velocities of other, as well as the positions of static obstacles. We also refer to both agents and static obstacles under the general term *obstacles* for this subsection, as agents are treated as moving obstacles whom we have no control over, where we utilize forwards dynamics to derive their future position based on their current velocity and heading. For any static obstacle j, we set $\theta^j = 0$ and $v^j = 0$. for static obstacles. Thus, from agent i's perspective, the transition dynamics are defined as

$$\dot{\mathbf{z}} = \begin{bmatrix} \mathbf{f}^i(\mathbf{x}^i) \\ \mathbf{f}^j(\mathbf{x}^j) \end{bmatrix} + \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}^i \\ \mathbf{u}^j \end{bmatrix}, \tag{3}$$

where $\dot{\mathbf{z}} = \left[\dot{x^i}, \dot{y^i}, \dot{\theta^i}, \dot{v^i}, \dot{x^j}, \dot{y^j}, \dot{\theta^j}, \dot{v^j}\right]^\top$. The control inputs are applied using an input matrix, $\mathbf{B}$, where $\mathbf{B} = [0, 0, 1, 0; 0, 0, 0, 1]^\top$, and the control vectors for agents i and j are $\mathbf{u}^i = \left[\omega^i, a^i\right]^\top$ and $\mathbf{u}^j = \left[\omega^j, a^j\right]^\top$, respectively. The dynamics for each agent are represented by $\mathbf{f}^i(\mathbf{x}^i)$ and $\mathbf{f}^j(\mathbf{x}^j)$, where $\mathbf{f}^i(\mathbf{x}^i) = \left[v^i \cos(\theta^i), v^i \sin(\theta^i), 0, 0\right]^\top$. For simplicity, agents and static obstacles are considered to occupy circles of their respective radius r. The barrier function for obstacle avoidance can be written out as $b(z) = (x^i - x^j)^2 + (y^i - y^j)^2 - (r^i + r^j)^2 >= 0$ where $(x^j, y^j) \in \mathbb{R}^2$ represents the position of obstacle j, and r^j represents the radius of obstacle j.

As shown in (Xiao et al. (2023)), the HOCBF for $b(z)$ which has degree 2 with respect to the control outputs yields the following inequality:

$$-L_f L_g b(z) u \leq L_f^2 b(z) + (p_1^o(z) + p_2^o(z)) L_f b(z) + (\dot{p_1^o}(z) + p_1^o(z) p_2^o(z)) b(z), \tag{4}$$

where $p_1^o(z)$ and $p_2^o(z)$ are trainable penalty functions, and $\dot{p_1^o}(z)$ can be set to 0 due to the discretization solving method of the QP as shown by (Xiao et al. (2023)). Additionally, we use sigmoid as a continuous and differentiable activation function for the penalty network to ensure that $p_1^o(z)$ and $p_2^o(z)$ are Lipschitz continuous, maintaining BarrierNet's safety guarantees. Given that $b(z) = (x^i - x^j)^2 + (y^i - y^j)^2 - (r^i + r^j)^2$, we can solve for the Lie derivatives of $b(z)$ in the $f(\mathbf{x})$ and $g(\mathbf{x})$ vector fields:

$$L_f b(z) = 2(x^i - x^j)(v^i \cos(\theta^i) - v^j \cos(\theta^j)) + 2(y^i - y^j)(v^i \sin(\theta^i) - v^j \sin(\theta^j)), \tag{5a}$$

$$L_f^2 b(z) = 2({v^i}^2 + {v^j}^2 - 2 v^i v^j (\cos(\theta^i + \theta^j)), \tag{5b}$$

$$L_g L_f b(z) = \begin{bmatrix} -2(x^i - x^j) v^i \sin(\theta^i) + 2(y^i - y^j) v^i \cos(\theta^i) \\ 2(x^i - x^j) \cos(\theta^i) + 2(y^i - y^j) \sin(\theta^i) \end{bmatrix}^T. \tag{5c}$$

These values, plugged into Equation (4), generate our differential CBF constraint.

4.2. Minimally Invasive Deadlock Prevention dCBFs

We introduce a minimally invasive differential CBF layer to look ahead and output accelerations that avoid collisions and maintain liveness in SMG scenarios. We first check if agent i's projected spatial path intersects with the projected spatial path of any other agent j for all j in $[1; k]$ s.t. $i \neq j$, assuming that agent j maintains their heading (θ^j) and velocity (v^j). This check boils down to a ray intersection, which occurs when the following is satisfied:

$$(\Delta y * \cos(\theta^i) - \Delta x * \sin(\theta^i)) * det > 0, \qquad (\Delta y * \cos(\theta^j) - \Delta x * \sin(\theta^j)) * det > 0, \tag{6}$$

where $det = \hat{v^j}_x * \hat{v^i}_y - \hat{v^j}_y * \hat{v^i}_x$ and $\hat{v^i} = \frac{v^i}{|v^i|}, \hat{v^j} = \frac{v^j}{|v^j|}$. The minimally invasive liveness filter is only applied if the rays intersect.

Let $\widetilde{s^i} \in C^i$ be the closest point on agent i's convex hull to agent i. Similarly, let $\widetilde{s^j} \in C^j$ be the closest point on agent j's convex hull to agent i. We denote $\widetilde{s^i}$ and $\widetilde{s^j}$ to be the agents' *critical points*. Assuming that the agents have not yet collided, these points lie on the boundaries of their respective agent's convex hull, and thus are r^i and r^j from the agents' current positions. Mathematically,

$$\widetilde{s^i} = \hat{s}^i + r^i * \hat{s}^{j-i}, \qquad \widetilde{s^j} = \hat{s}^j - r^j * \hat{s}^{j-i}, \tag{7}$$

where $\hat{s}^{j-i} = \frac{s^j - s^i}{|s^j - s^i|}$. Thus, our problem reduces to avoiding a point-point collision instead of a convex-convex collision. We first calculate c, the potential collision point of $\widetilde{s^i}$ and $\widetilde{s^j}$, by projecting them forwards along the agents' current spatial path given θ and v:

$$\begin{aligned} &\widetilde{s^i}' = \widetilde{s^i} + \mathbf{v}^i, && \widetilde{s^j}' = \widetilde{s^j} + \mathbf{v}^j, \\ &a^i = \widetilde{x}^i * \widetilde{y}^{i'} - \widetilde{y}^i * \widetilde{x}^{i'}, && a^j = \widetilde{x}^j * \widetilde{y}^{j'} - \widetilde{y}^j * \widetilde{x}^{j'}, \\ &c_x = (a^j v^i \cos(\theta^i) - a^i v^j \cos(\theta^j))/det, && c_y = (a^j v^i \sin(\theta^i) - a^i v^j \sin(\theta^j))/det, \end{aligned} \tag{8}$$

where $det = v^i v^j(\cos(\theta^i)\sin(\theta^j) - \sin(\theta^i)\cos(\theta^j))$. We then calculate the distance from each agent's critical point $\widetilde{s}$ to the collision point.

$$d^i = \sqrt{(\widetilde{x}^i - c_x)^2 + (\widetilde{y}^i - c_y)^2}, \qquad d^j = \sqrt{(\widetilde{x}^j - c_x)^2 + (\widetilde{y}^j - c_y)^2}. \tag{9}$$

Given each agent's velocity, we calculate t^i and t^j, the time for each agent's critical point, $\widetilde{s}$, to reach the collision point c as $t = v/d$. We split the scenario into two different cases. If $t^i < t^j$, then that indicates that the ego agent, agent i, will pass the collision point before agent j. In this scenario, to maintain liveness we want to enforce that $t^j > t^i$, thus resulting in the barrier function

$$b(z) = t^j - t^i \geq 0. \tag{10}$$

As stated previously, $t^j = \frac{d^j}{v^j}$ and $t^i = \frac{d^i}{v^i}$. Since the c lies along each agent's heading, θ, and since a minimally invasive trajectory involves zero spatial deviation from the desired path, v is the direct derivative of d. That is, $dd/dt = v$. Since our barrier function is with respect to v^i, and the control input a^i appears in the first derivative of our barrier function, we alter the HOCBF inequality from Equation (4) to instead be

$$-L_g b(z) u \leq L_f b(z) + p^l b(z). \tag{11}$$

Thus, we compute the Lie derivatives and formulate the CBF in Equation (12).

$$b(z) = \delta(t^j - t^i), \qquad L_g b(z) = \delta\left(\frac{a^i d^i}{{v^i}^2}\right), \qquad -\delta\left(\frac{a^i d^i}{{v^i}^2}\right) \leq p^l_\delta(x)\left(\frac{d^j}{v^j} - \frac{d^i}{v^i}\right), \tag{12}$$

where $\delta(\cdot)$ is an indicator function and is equal to 1 if $t^i < t^j$ (the ego-agent will currently reach sooner) and -1 if $t^i >= t^j$ (the ego-agent will currently reach later and should yield to agent j). Note that there is no Lie derivative along the f function since in the absence of any control outputs, the barrier function $t^j - t^i$ would remain constant over time. The penalty value, $p^l_\delta(z)$ is chosen for these CBFs to allow the network to learn the necessary constraint levels in each scenario.

Method	# Collisions	# Deadlocks	Makespan (s)	ΔV (m/s)	Δ Path (m)	Cycle Time (s)
Doorway Scenario						
MPC-CBF (Zeng et al. (2021))	0	50	N/A	0.003 ± 0.000	0.016 ± 0.000	90.9 ± 0.9
MACBF (Qin et al. (2021))	50	0	N/A	0.006 ± 0.000	0.149 ± 0.048	171.05 ± 1.66
PIC (Liu et al. (2020))	50	0	N/A	0.031 ± 0.006	0.041 ± 0.002	0.3 ± 0.0
BarrierNet (Xiao et al. (2023))	50	0	N/A	0.004 ± 0.000	0.010 ± 0.002	7.3 ± 0.0
SMG-CBF (Chandra et al. (2024))	0	0	13.8 ± 0.0	0.009 ± 0.000	0.001 ± 0.000	81.1 ± 0.3
LIVENET	0	0	13.8 ± 0.0	0.002 ± 0.000	0.008 ± 0.000	7.5 ± 0.0
Intersection Scenario						
MPC-CBF (Zeng et al. (2021))	0	50	N/A	0.006 ± 0.000	0.170 ± 0.001	302.6 ± 3.9
MACBF (Qin et al. (2021))	50	0	N/A	0.300 ± 0.002	0.009 ± 0.004	170.8 ± 1.6
PIC (Liu et al. (2020))	50	0	N/A	0.081 ± 0.024	0.033 ± 0.003	0.3 ± 0.0
BarrierNet (Xiao et al. (2023))	50	0	N/A	0.008 ± 0.000	0.033 ± 0.001	7.3 ± 0.0
SMG-CBF (Chandra et al. (2024))	0	0	12.2 ± 0.0	0.012 ± 0.000	0.000 ± 0.000	157.0 ± 0.5
LIVENET	0	0	11.6 ± 0.0	0.011 ± 0.000	0.000 ± 0.000	9.3 ± 0.1

Table 2: Experiment results in the Doorway and Intersection scenarios, averaged over 50 runs.

Theorem 1 *Assuming $p^l_{-1}(z), p^l_1(z)$ are differentiable functions with respect to z, then the* LIVENET *constraints in Equation* (12) *guarantee the liveness of the system defined by* (10).

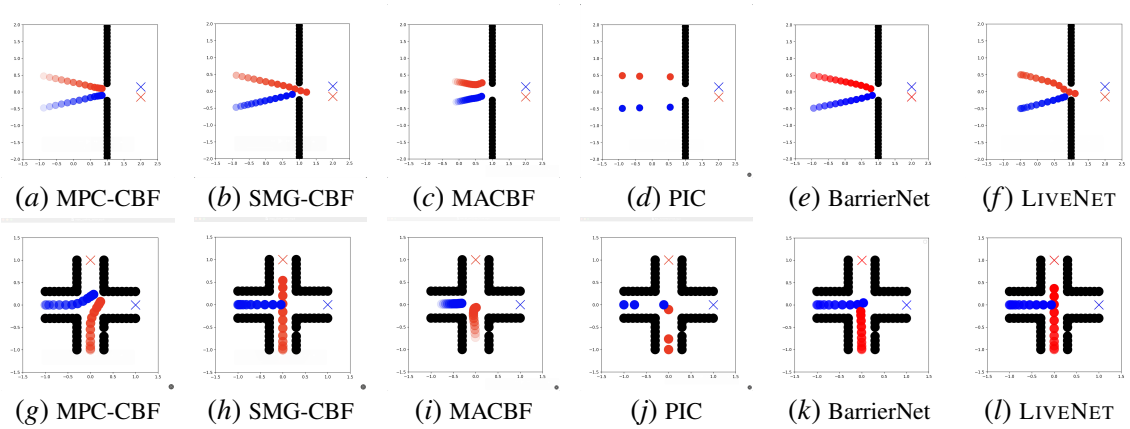
Proof The proof follows directly from (Xiao et al. (2023))(c.f. Theorem 2) and relies on the requirement that the relative degree of $p^l_{-1}(z), p^l_1(z)$ with respect to each component in z is greater than or equal to that of the liveness constraints in Equations (12). In Equations (12), since the control input (a^i) appears in the first derivative, the relative degree is 1. Next, recall that each agent i has a partial observability over the positions and velocities of other robots in its neighborhood, that is, $z = [s^i, v^i, s^j, v^j]$. The penalty functions $p^l_{-1}(z), p^l_1(z)$ have a relative degree of 2 with respect to position and 1 with respect to velocity, therefore, the conditions set forth in (Xiao et al. (2023)) are satisfied. ■

5. Experiments and Results

We aim to investigate two main questions: (i) how does LIVENET compare with existing multi-robot navigation methods in SMGs? and (ii) how does LIVENET compare with methods specifically designed to navigate SMGs? To investigate these questions, we compare LIVENET to five baseline methods. These baselines include two receding-horizon optimization-based controllers, MPC-CBF (Zeng et al. (2021)), which maintains safety from static and moving obstacles using CBFs and SMG-CBF (Chandra et al. (2024)), an extension of the MPC-CBF formulation that employs a threshold-based liveness CBF with steep acceleration outputs. Due to the deterministic nature of SMG-CBF, it was predetermined which agent would go faster in perfectly symmetric scenarios. Two multi-agent learning-based approaches were also tested: MACBF (Qin et al. (2021)), which learns safety through separated action and CBF networks, and PIC (Liu et al. (2020)), which utilizes graph CNNs with a permutation invariant critic for scalable navigation. BarrierNet (Xiao et al. (2023)) was also tested as a baseline. We measure number of collisions and deadlocks, makespan, and runtime per control iteration (in seconds) in a variety scenarios. To evaluate minimal invasiveness between the different approaches, we measure average change in velocity and the average deviation from the desired path.

5.1. Experiment Setup

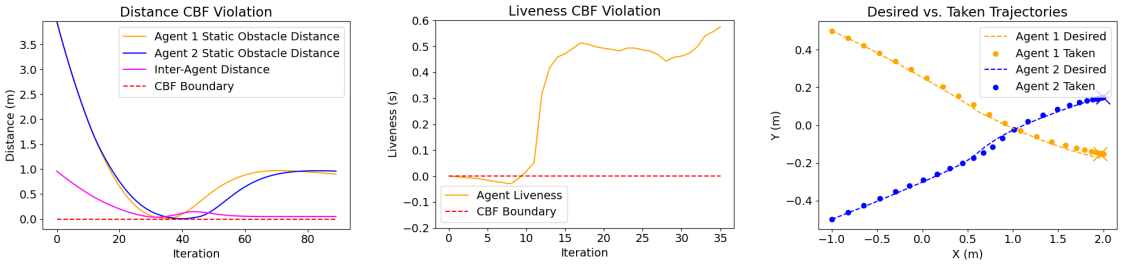
The simulation environment was setup in Python using the `do_mpc` framework (Lucia et al. (2017)) based on Casadi (Andersson et al. (2019)). The kindoynamic and scenario time constraints remained constant across all scenarios: max velocity of $0.3\ m/s$, max acceleration / deceleration of $0.1\ m/s^2$, max angular velocity of $0.5\ rad/s$, agent radii of $0.1\ m$, sim fidelity of $0.2\ s/iteration$, and max sim time of $18s$. The following symmetric SMGs were tested: (i) *Doorway scenario*: multiple robots pass through a doorway $0.3m$ wide at $(1, 0)$ wide. The agents are placed $(-1, \pm 0.5)$ with



(*a*) MPC-CBF (*b*) SMG-CBF (*c*) MACBF (*d*) PIC (*e*) BarrierNet (*f*) LIVENET

(*g*) MPC-CBF (*h*) SMG-CBF (*i*) MACBF (*j*) PIC (*k*) BarrierNet (*l*) LIVENET

Figure 4: Resulting trajectories in Doorway (Figures 4(*a*)-4(*f*)) and Intersection (Figures 4(*g*)-4(*l*))



(*a*) Distance CBF (p^o) (*b*) Liveness CBF (p^l) (*c*) Average Δ Path

Figure 5: LIVENET's obstacle dCBF, liveness dCBF, and deviation from desired path in Doorway scenario.

initial velocity $0.3m/s$. *Intersection scenario*: multiple robots cross a $0.35m \times 0.35m$ intersection. The robots started $1m$ away from the intersection with their goal placed $1m$ past the intersection.

Training: To train LIVENET, we generated data using an optimal, receding-horizon MPC-CBF controller (Zeng et al. (2021)) across various perturbations of the doorway and intersection scenarios. Parameters such as starting state, goal state, and gap size were perturbed to increase diversity. For each scenario perturbation, the MPC's state and input cost matrices, as well as the CBF parameters, were individually tuned to generate an optimal trajectory. LIVENET was subsequently trained on this augmented data through offline supervised learning, using mean squared error as the loss function. The training process spanned 30 epochs of shuffled data, with a batch size of 64 and a learning rate of 0.001. Baseline methods (Figure 2) were trained in the same environment, with the exception of PIC, which performed better when trained within its native training environment and was subsequently adapted to the SMG environment through a custom state-action mapping.

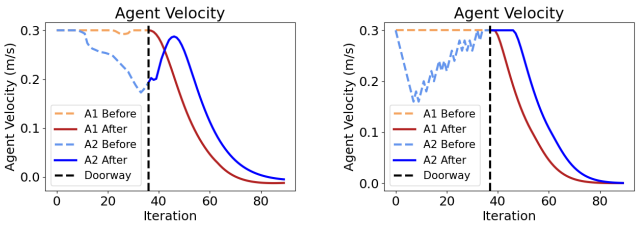
5.2. Results

For each agent, 50 Doorway and Intersection scenarios were run to test the safety, liveness, and smoothness of the trajectories of each agent. The accumulated and averaged metric values are displayed in Table 2 and the resulting trajectories are shown in Figure 4. The MACBF, PIC, and BarrierNet models resulted in collisions due to mere soft constraints on safety with the limited training data. The MPC-CBF model was able to avoid collisions, but succumbed to deadlocks due to an inability to make safe progress. Additionally, LIVENET performed minimally invasive behavior as it maintained its desired spatial trajectory and minimally perturbed its velocity, approaching the CBF threshold without ever violating it (Figure 5).

Of the baselines, only SMG-CBF was successful due to its deadlock resolution capabilities. Additionally, when SMG-CBF was tested without the predetermination of which agent would be faster, it resulted in deadlocks in perfectly symmetric versions of both the Doorway and Intersection

scenarios. SMG-CBF, however, is $10\times$ slower than LIVENET as it runs an iterative optimization algorithm every control cycle. Additionally, as shown in the Doorway scenario results, SMG-CBF was significantly more invasive, with an average velocity perturbation of 4–$5\times$ that of LIVENET. Another core limitation with SMG-CBF is its dependence on a *liveness threshold* to determine when to apply the CBF (Chandra et al. (2024)). As the controller enters and exits this threshold, its acceleration output varies significantly, thus producing jagged controls as shown in Figure 6(*b*) between iterations 10 and 40. On the other hand, LIVENET's ability to learn how much to relax the CBF as a function of the agent's state and observation allows the velocity profile of the resultant path to be much smoother as demonstrated by the smoother dip in agent 2's velocity in Figure 6(*a*), which more closely mimics human-like behavior. Figure 6 further shows evidence of better human-like yielding for LIVENET. In particular, note that agent 2 does not begin to yield until the last second (around iteration 20) compared to SMG-CBF where agent 2 begins to slow around iteration 10, suggesting that LIVENET results in less conservative, more agile navigation.

LIVENET is also more robust to variations in the environment and agent configurations thanSMG-CBF. We tested both on a suite of 28 perturbed scenarios of the original doorway SMG without any changes to their parameter configuration. The perturbations were created by variations in the agents' initial position, initial heading, initial velocity, and goal position. The positions were altered on a scale of $0.5m$, initial headings facing the doorway and facing the wall were tested, and the initial velocity was either full speed ($0.3m/s$) or standstill ($0.0m/s$). LIVENET was able to solve 25 / 28 scenarios without a deadlock or collision, whereas SMG-CBF was only able to solve 16 / 28. It should also be noted that due to SMG-CBF's deterministic manner, it is unable to solve perfectly symmetrical cases without predefining which agent should start off moving faster. On the other hand, the same LIVENET network could be used for both agents, as it has a slight inherent bias based on its starting and goal positions, allowing it to break the symmetry. SMG-CBF's lack of robustness is due to constant parameters defining how strictly the CBF is followed for each scenario. On the other hand, LIVENET's ability to predict the penalty values, $p(z)$, that define the relaxation of the CBF allows it to better adapt to a multitude of scenarios.



(*a*) LIVENET (*b*) SMG-CBF

Figure 6: Comparing agents' (A1, A2) velocities generated by LIVENET and SMG-CBF, before and after crossing the doorway.

6. Conclusion

In this work, we presented LIVENET, a robust, minimally-invasive neural network controller that uses differentiable CBF layers to tackle safety and liveness in constrained environments. Our navigation approach utilized the BarrierNet framework as the base neural network. We introduced novel differentiable CBF layers to provide liveness and 2D multi-agent navigation. To train the network, we hand-tuned an optimal recending-horizon controller over many perturbed scenarios to generate a large dataset. Our approach guarantees safe and live behavior given enough training data and a complex enough network to learn the dCBF's corresponding penalty values. Experiments show that in practical scenarios the model outperforms existing solutions in safety, minimal invasiveness, compute speed, and robustness. The faster compute time and robustness are crucial when run on real robots in constrained areas that need to react quickly to unpredictable situations.

Our approach has some limitations. LIVENET is currently tested in simulation, and we plan to deploy it into the real world in the future. Additionally, LIVENET has only been tested on 2-agent scenarios. We plan on investigating the scalability of this model in terms of the compute time and accuracy as the number of agents increases. Additionally, since LIVENET is a discrete controller, collisions and deadlocks can occur between timesteps. Furthermore, an issue with imitation learning is a laborious data generation process, as we have to tune the optimal controller for each scenario perturbation. Utilizing unsupervised learning methods would allow for self-exploration of novel states instead of forcing the agent to only learn from states that the optimal controller explored.

References

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pages 136–145. PMLR, 2017.

Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11:1–36, 2019.

Tsz-Chiu Au, Shun Zhang, and Peter Stone. Autonomous intersection management for semi-autonomous vehicles. In *Routledge Handbook of Transportation*, pages 88–104. Routledge, 2015.

Robert Bogue. The role of robots in logistics. *Industrial Robot: the international journal of robotics research and application*, 51(3):381–386, 2024.

Dustin Carlino, Stephen D Boyles, and Peter Stone. Auction-based autonomous intersection management. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 529–534. IEEE, 2013.

Rohan Chandra, Vrushabh Zinage, Efstathios Bakolas, Peter Stone, and Joydeep Biswas. Deadlock-free, safe, and decentralized multi-robot navigation in social mini-games via discrete-time control barrier functions. 2024.

Yuda Chen, Chenghan Wang, Meng Guo, and Zhongkui Li. Multi-robot trajectory planning with feasibility guarantee and deadlock resolution: An obstacle-dense environment. *IEEE Robotics and Automation Letters*, 8(4):2197–2204, 2023.

Shreyansh Daftry, J Andrew Bagnell, and Martial Hebert. Learning transferable policies for monocular reactive mav control. In *2016 International Symposium on Experimental Robotics*, pages 3–11. Springer, 2017.

Bolei Di and Andrew Lamperski. Differential dynamic programming for nonlinear dynamic games. *arXiv preprint arXiv:1809.08302*, 2018.

David Fridovich-Keil, Ellis Ratner, Lasse Peters, Anca D Dragan, and Claire J Tomlin. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 1475–1481. IEEE, 2020.

Kunal Garg, Songyuan Zhang, Jacob Arkin, and Chuchu Fan. Foundation models to the rescue: Deadlock resolution in connected multi-robot systems, 2024. URL https://arxiv.org/abs/2404.06413.

Jaskaran Grover, Changliu Liu, and Katia Sycara. The before, during, and after of multi-robot deadlock. *The International Journal of Robotics Research*, 42(6):317–336, 2023.

Jianhua Guo, Zhihao Xie, Ming Liu, Zhiyuan Dai, Yu Jiang, Jinqiu Guo, and Dong Xie. Spatio-temporal joint optimization-based trajectory planning method for autonomous vehicles in complex urban environments. *Sensors*, 24(14):4685, 2024.

Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.

Ahmed Hussein, Eyad Elyan, Mohamed Medhat Gaber, and Chrisina Jayne. Deep imitation learning for 3d navigation tasks. *Neural computing and applications*, 29:389–404, 2018.

Iou-Jen Liu, Raymond A Yeh, and Alexander G Schwing. Pic: permutation invariant critic for multi-agent deep reinforcement learning. In *Conference on Robot Learning*, pages 590–602. PMLR, 2020.

Sergio Lucia, Alexandru Tătulea-Codrean, Christian Schoppmeyer, and Sebastian Engell. Rapid development of modular and sustainable nonlinear model predictive control solutions. *Control Engineering Practice*, 60:51–62, 2017. ISSN 0967-0661. doi: https://doi.org/10.1016/j.conengprac.2016.12.009. URL https://www.sciencedirect.com/science/article/pii/S0967066116302970.

Hang Ma. Graph-based multi-robot path finding and planning. *Current Robotics Reports*, 3(3): 77–84, 2022.

Katherine Mao, Igor Spasojevic, Malakhi Hopkins, M Ani Hsieh, and Vijay Kumar. Collision-free time-optimal path parameterization for multi-robot teams. *arXiv preprint arXiv:2409.17079*, 2024.

Francisco Martinez-Gil, Miguel Lozano, and Fernando Fernández. Multi-agent reinforcement learning for simulating pedestrian navigation. In *Adaptive and Learning Agents: International Workshop, ALA 2011, Held at AAMAS 2011, Taipei, Taiwan, May 2, 2011, Revised Selected Papers*, pages 54–69. Springer, 2012.

Matthew McNaughton, Chris Urmson, John M Dolan, and Jin-Woo Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *2011 IEEE International Conference on Robotics and Automation*, pages 4889–4895. IEEE, 2011.

Negar Mehr, Mingyu Wang, Maulik Bhatt, and Mac Schwager. Maximum-entropy multi-agent dynamic games: Forward and inverse solutions. *IEEE transactions on robotics*, 39(3):1801–1815, 2023.

Pol Mestres, Carlos Nieto-Granda, and Jorge Cortés. Distributed safe navigation of multi-agent systems using control barrier function-based controllers. *IEEE Robotics and Automation Letters*, 2024.

Jun Morimoto and Christopher G Atkeson. Minimax differential dynamic programming: An application to robust biped walking. In *Advances in neural information processing systems*, pages 1563–1570, 2003.

Thulasi Mylvaganam, Mario Sassano, and Alessandro Astolfi. A differential game approach to multi-agent collision avoidance. *IEEE Transactions on Automatic Control*, 62(8):4229–4235, 2017. doi: 10.1109/TAC.2017.2680602.

Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. Learning safe multi-agent control with decentralized neural barrier certificates. *CoRR*, abs/2101.05436, 2021. URL https://arxiv.org/abs/2101.05436.

Ammar Abdul Ameer Rasheed, Mohammed Najm Abdullah, and Ahmed Sabah Al-Araji. A review of multi-agent mobile robot systems applications. *International Journal of Electrical and Computer Engineering*, 12(4):3517–3529, 2022.

Wilko Schwarting, Alexander Pierson, Sertac Karaman, and Daniela Rus. Stochastic dynamic games in belief space. *IEEE Transactions on Robotics*, pages 1–16, 2021.

Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial intelligence*, 195:470–495, 2013.

Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial intelligence*, 219:40–66, 2015.

Roni Stern. Multi-agent path finding–an overview. *Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures*, pages 96–115, 2019.

Wenliang Sun, Evangelos A Theodorou, and Panagiotis Tsiotras. Game theoretic continuous time differential dynamic programming. In *2015 American Control Conference (ACC)*, pages 5593–5598. IEEE, 2015.

Wenliang Sun, Evangelos A Theodorou, and Panagiotis Tsiotras. Stochastic game theoretic trajectory optimization in continuous time. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6167–6172. IEEE, 2016.

Nilesh Suriyarachchi, Rohan Chandra, John S Baras, and Dinesh Manocha. Gameopt: Optimal real-time multi-agent planning and control for dynamic intersections. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2599–2606. IEEE, 2022.

Li Wang, Aaron D Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3):661–674, 2017.

Mingyu Wang, Zijian Wang, John Talbot, J. Christian Gerdes, and Mac Schwager. Game-theoretic planning for self-driving cars in multivehicle competitive scenarios. *IEEE Transactions on Robotics*, 37(4):1313–1325, 2021a. doi: 10.1109/TRO.2020.3047521.

Mingyu Wang, Zijian Wang, John Talbot, J Christian Gerdes, and Mac Schwager. Game-theoretic planning for self-driving cars in multivehicle competitive scenarios. *IEEE Transactions on Robotics*, 37(4):1313–1325, 2021b.

Wei Xiao and Calin Belta. Control barrier functions for systems with high relative degree. In *2019 IEEE 58th conference on decision and control (CDC)*, pages 474–479. IEEE, 2019.

Wei Xiao, Tsun-Hsuan Wang, Ramin Hasani, Makram Chahine, Alexander Amini, Xiao Li, and Daniela Rus. Barriernet: Differentiable control barrier functions for learning of safe robot control. *IEEE Transactions on Robotics*, 39(3):2289–2307, 2023.

Chengzhen Yan, Jiahu Qin, Qingchen Liu, Qichao Ma, and Yu Kang. Mapless navigation with safety-enhanced imitation learning. *IEEE Transactions on Industrial Electronics*, 70(7):7073–7081, 2022.

Jun Zeng, Bike Zhang, and Koushil Sreenath. Safety-critical model predictive control with discrete-time control barrier function. In *2021 American Control Conference (ACC)*, pages 3882–3889. IEEE, 2021.

Dingjiang Zhou, Zijian Wang, Saptarshi Bandyopadhyay, and Mac Schwager. Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. *IEEE Robotics and Automation Letters*, 2(2):1047–1054, 2017.

Edward L Zhu, Yvonne R Stürz, Ugo Rosolia, and Francesco Borrelli. Trajectory optimization for nonlinear multi-agent systems using decentralized learning model predictive control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 6198–6203. IEEE, 2020.

Hai Zhu, Bruno Brito, and Javier Alonso-Mora. Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware voronoi cells. *CoRR*, abs/2201.04012, 2022. URL https://arxiv.org/abs/2201.04012.