Co-Optimizing Reconfigurable Environments and Policies for Decentralized Multi-Agent Navigation

Zhan Gao, *IEEE Student Member*, Guang Yang, *IEEE Member* and Amanda Prorok, *IEEE Senior Member*

arXiv:2403.14583v2 [cs.RO] 2 Jul 2025

***Abstract*—This work views the multi-agent system and its surrounding environment as a co-evolving system, where the behavior of one affects the other. The goal is to take both agent actions and environment configurations as decision variables, and optimize these two components in a coordinated manner to improve some measure of interest. Towards this end, we consider the problem of decentralized multi-agent navigation in a cluttered environment, where we assume that the layout of the environment is reconfigurable. By introducing two sub-objectives—multi-agent navigation and environment optimization—we propose an *agent-environment co-optimization* problem and develop a *coordinated algorithm* that alternates between these sub-objectives to search for an optimal synthesis of agent actions and environment configurations; ultimately, improving the navigation performance. Due to the challenge of explicitly modeling the relation between the agents, the environment and their performance therein, we leverage policy gradient to formulate a model-free learning mechanism within the coordinated framework. A formal convergence analysis shows that our coordinated algorithm tracks the local minimum solution of an associated time-varying non-convex optimization problem. Experiments corroborate theoretical findings and show the benefits of co-optimization. Interestingly, the results also indicate that optimized environments can offer structural guidance to de-conflict agents in motion.**

***Index Terms*—Coordinated optimization, environment, multi-agent system, decentralized navigation.**

I. Introduction

Multi-agent systems comprise multiple decision-making agents that interact in a shared environment to achieve common or individual goals. They present an attractive solution to spatially distributed tasks, wherein efficient and safe motion planning is one of the central problems. Classically, the field of multi-agent motion planning does not consider the surrounding environment as a decision variable, instead acquiring environmental information through perceptual techniques and modeling it as an unmodifiable constraint [1]–[6]. However, such spatial constraints may result in deadlocks, live-locks and prioritization conflicts even for state-of-the-art algorithms [7], [8], highlighting the environment impact on agents' performance. Yet, despite the entanglement of these two components, coordinated optimization of multi-agent policies and environment configurations has received little attention thus far.

With advances in programmable and responsive buildings, reconfigurable and automated environments are emerging as a new trend [9]–[12]. For example, the locations of shelves in warehouses can be adjusted with rack pulleys or sliding track systems; the layout of manufacturing equipment in factories can be re-designed or re-arranged within pre-designated regions; and the positions of dining tables in restaurants can be adapted to better suit food delivery paths. In parallel to developing agent policies, this means that we have the opportunity to re-configure the spatial layout of the environment as a means to improve some objective performance measure. The latter is especially appealing when agents are expected to solve repetitive tasks in structured settings (like warehouses, factories, restaurants, etc.). While it is possible to hand-design environment configurations, such a procedure is inefficient and often sub-optimal [13].

Recent works have suggested that significant benefits can be reaped by either adapting the environment to agent policies or tuning agent policies based on the environment [14]–[19]. These insights motivate the coupling of the environment configuration design with the agents' policy development. The goal of this work is to consider the obstacle layout of the environment as a decision variable, as well as the navigation policy of agents, and propose a system-level co-optimization problem that simultaneously optimizes two components; ultimately, *establishing a symbiotic agent-environment system* that maximizes the navigation performance of the multi-agent system. Applications include warehouse logistics (e.g., finding the optimal shelf positions and robot policies for cargo transportation [20]), search and rescue in collapsed sites (e.g., clearing passageways and deploying rescue strategies for trapped victims [21]), city planning (e.g., designing the optimal one-way routes and autonomous vehicle strategies for traffic efficiency [22]), and digital entertainment (e.g., building gaming scenes that elicit the best possible behaviors in animated video game characters [23]).

In pursuit of this goal, we propose a *coordinated optimization* algorithm that alternates between the synthesis of navigation behavior and environment design. Due to the challenge of explicitly modeling the relation between agents, environment and navigation performance, we pose the problem in a model-free learning framework and introduce two sub-objectives of multi-agent navigation and environment optimization, respectively. We parameterize the navigation policy of agents with a graph neural network (GNN) for decentralized implementation and use a generative model of the environment, parameterized by a deep neural network (DNN). The two models are updated with respect to two sub-objectives in an alternating manner, pursuing an optimal pair of navigation and generative parameters that maximizes the multi-agent system performance – see Fig. 1 for an overview of the agent-environment coordinated optimization framework. In more detail, our contributions are:

The authors are with the Department of Computer Science and Technology, University of Cambridge, UK (Email: {zg292, asp45}@cam.ac.uk, gyang101@bu.edu). This work is supported by ERC Project 949940 (gAIa).

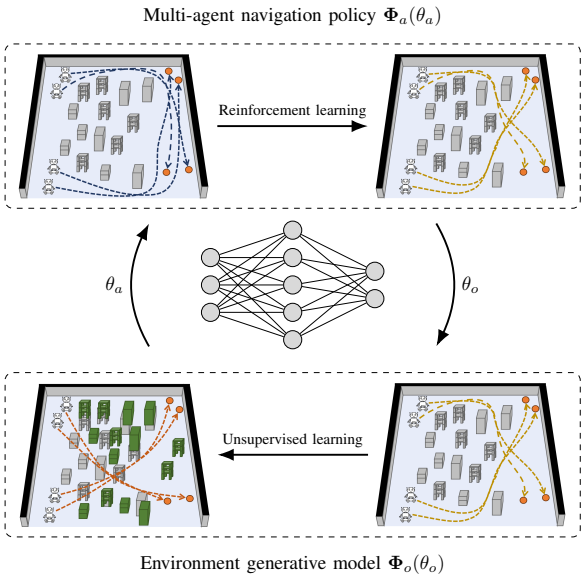


Figure 1. Agent-environment co-optimization is an optimization process that alternates between a multi-agent navigation policy (parameterized by θ_a) and an environment generative model (parameterized by θ_o), searching for an optimal pair (θ_a^*, θ_o^*) maximizing the system performance. In the above example, it optimizes the navigation policy with reinforcement learning (e.g., blue trajectories become yellow ones) and the environment configuration with unsupervised learning (e.g., grey obstacles become green ones), in an alternating manner.

- We propose the problem of agent-environment co-optimization, which considers both environment configurations and multi-agent policies as decision variables in a system-level optimization framework. The goal is to search for an optimal pair that jointly maximizes the navigation performance.
- We develop a coordinated method that optimizes the navigation policy of the agents with reinforcement learning and a generative model of the environment configuration with unsupervised learning, based on two sub-objectives, in an alternating manner. We formulate a model-free learning mechanism within the coordinated framework, which overcomes the challenge of modeling the explicit relation between agents, environment and performance.
- We provide a formal convergence analysis for the proposed coordinated method by detailing its relation with an associated non-convex time-varying optimization problem and an ordinary differential equation.
- We perform simulations as well as real-world experiments to evaluate the proposed method. The results corroborate theoretical findings and provide novel insights regarding the relationship between the environment and the multi-agent system. That is, we show how appropriately designed environments are able to guide multi-agent navigation through targeted spatial de-confliction.

Related work. The idea of co-optimization (or co-design) in robotics can be traced back to work on *embodied cognitive science* [24], which emphasizes the importance of the robot body and its interaction with the environment for achieving robust behavior. Motivated by this idea, research in the domain of robot co-design aims to find an optimized controller, sensing and morphology, such that they together produce the desired performance [25]–[27]. A few seminal works tackled this problem with an evolutionary approach that co-optimizes objectives [28], [29]. More recently, [30] employed reinforcement learning to learn both the policy and morphology design parameters in an alternating manner, where a single shared policy controls the distribution of different robot designs. Within the multi-agent domain, the concept of co-optimization has mainly been leveraged to allocate physical resources to agent tasks. The work in [31] proposed a joint equilibrium policy search method that encourages agents to cooperate to reach states maximizing a global reward, while [32] developed a multi-level search approach that co-optimizes agent placements with task assignment and scheduling. In a similar vein, [33] used online multi-agent reinforcement learning to co-optimize cores, caches and on-chip networks. The works in [34], [35] considered the problem of co-optimizing mobility and communication among agents, and designed schedule policies that minimize the total energy. While broadly similar in their ideas, none of these works consider co-optimizing the *environment* and *navigation behaviors* to improve the system performance of multiple agents (i.e., robots).

The key challenge of co-optimization stands in the fact that the explicit relation between agents, environment and system performance is generally not known (and hard to model), compounding the difficulty of solving the joint problem. The agent-environment relationship has been previously explored. The works in [36], [37] identified the existence of congestion and deadlocks in undesirable environments and developed trajectory planning methods for agents to escape from these potential deadlocks, while [16] introduced the concept of "well-formed" environment where the navigation tasks of all agents can be carried out successfully without collision. Wu et al. [4] showed that the environment shape can result in distinct path prospects for different agents and proposed to leverage this information for coordinating agents' motion. Additionally, [38] focused on adversarial environments and developed resilient navigation algorithms for agents in these environments. However, none of the aforementioned works consider *both* the environment as well as the agents' policies as decision variables in a system-level optimization problem (with the aim of improving navigation performance).

More in the vein of our work, Hauser [39], [40] attempted to remove obstacles from the environment to improve the agent's navigation performance, while focusing on a single agent scenario. The work in [11] extended a related idea to the multi-agent setting, to search over all possible environment configurations to find the best solution for agents. However, the technique is limited to discrete settings, and only considers obstacle removal (in contrast to general re-configuration). The problem of multi-agent pick-up and delivery also explores the relationship between agents and environment [41]–[43]. However, it considers environment changes as pre-defined tasks and assigns agents to complete these tasks – constituting a different problem that is defined only in a discrete setting.

II. Problem Formulation

Let $\mathcal{E}$ be a 2-D environment with m obstacles $\mathcal{O} = \{O_j\}_{j=1}^m$. The obstacles are dispersed within the obstacle region Δ

according to specific environment configurations. Consider a multi-agent system with n agents $\mathcal{A}=\{A_i\}_{i=1}^n$ distributed in $\mathcal{E}$. The agents are initialized randomly in the starting region $\mathcal{S}$ and deploy a navigation policy π_a to move towards the goal region $\mathcal{D}$. We are interested in *decentralized* multi-agent navigation, where agents have access to their own states and exchange local observations with neighbors in the communication range. Our performance metrics include *traveled distance*, *average speed* and *collision avoidance*, which depend not only on the agents' policy π_a but also the surrounding environment $\mathcal{E}$. For example, we may facilitate the navigation by either developing an efficient policy, or designing a "well-formed" environment with an appropriate obstacle layout, or both.

This motivates a view of agents $\mathcal{A}$ and their environment $\mathcal{E}$ as a co-evolving system. The problem of *agent-environment co-optimization* then considers both components as decision variables for two objectives: *(i)* finding the optimal policy π_a^* for agents to reach goal positions without collision *(ii)* finding the optimal obstacle layout for the environment $\mathcal{E}^*$. The goal is to achieve these objectives simultaneously, i.e., *improving the navigation policy while, towards the same end, optimizing the obstacle layout*. This explores the coupling between the agents' policy and the environment configuration, to find an optimal pair $(\pi_a^*, \mathcal{E}^*)$ that maximizes the performance. In the following, we first introduce individual problem components and then formulate the co-optimization problem.

A. Multi-Agent Navigation

We consider the agents $\mathcal{A}$ initialized at starting positions $\mathbf{S} = [\mathbf{s}_1, ..., \mathbf{s}_n] \in \mathcal{S}$ and tasked towards goal positions $\mathbf{D} = [\mathbf{d}_1, ..., \mathbf{d}_n] \in \mathcal{D}$. Let $\mathbf{X}_a = [\mathbf{x}_{a1}, ..., \mathbf{x}_{an}]$ be the agent states and $\mathbf{U}_a = [\mathbf{u}_{a1}, ..., \mathbf{u}_{an}]$ the agent actions. For example, $\mathbf{X}_a$ are the positions or velocities and $\mathbf{U}_a$ are the accelerations. Let $\pi_a(\mathbf{U}_a|\mathbf{X}_a)$ be the navigation policy that generates actions $\mathbf{U}_a$ to steer the agents from $\mathbf{S}$ towards $\mathbf{D}$, which is a distribution of $\mathbf{U}_a$ conditioned on states $\mathbf{X}_a$. We model the navigation process as a *sequential* decision-making problem, where agents take actions based on their instantaneous states (e.g., positions, velocities and goals) across time steps t and do not require knowledge of the environment at run-time.

In this work, we consider the decentralized navigation problem where states are local observations (available to individual agents) and the full state is not observable [6], [44]–[47]. The decentralized setting is of practical importance because: (i) There may not exist a central processing unit to compute actions of all agents or each agent may not be able to obtain global information; (ii) it may be challenging to maintain a centralized condition throughout time; and (iii) the centralized setting introduces a single point of failure.

The navigation performance of the multi-agent system is determined by the navigation task, the moving capacity of the agents (e.g., maximal velocity) and the physical restrictions of the environment (e.g., obstacle blocking). In this context, we use an objective function $f(\mathbf{S}, \mathbf{D}, \pi_a, \mathcal{E})$ that depends on the initial positions $\mathbf{S}$, the goal positions $\mathbf{D}$, the navigation policy π_a, and the environment $\mathcal{E}$. The goal of multi-agent navigation is to find the optimal π_a^* that maximizes $f(\mathbf{S}, \mathbf{D}, \pi_a^*, \mathcal{E})$ given the navigation task $(\mathbf{S}, \mathbf{D})$ and the environment $\mathcal{E}$.

B. Environment Optimization

The obstacle layout $\mathcal{O}$ determines the environment configuration $\mathcal{E}$ for multi-agent navigation. Let $\pi_o(\mathcal{O}|\mathbf{S}, \mathbf{D})$ be the generative model that configures the obstacle layout based on the navigation task, i.e., agents' initial and goal positions, which is a distribution of $\mathcal{O}$ conditioned on $(\mathbf{S}, \mathbf{D})$ of the agents $\mathcal{A}$. The outputs of the generative model are re-configurable parameters of the obstacle layout, e.g., the obstacle position and the obstacle size, which are the degrees of freedom for environment optimization.

Since the environment $\mathcal{E}$ is determined by the obstacle layout and the latter is determined by the generative model π_o, we can re-write the objective function as $f(\mathbf{S}, \mathbf{D}, \pi_a, \pi_o)$. The goal of environment optimization is to find the optimal π_o^* that generates the environment $\mathcal{E}^*$ to maximize $f(\mathbf{S}, \mathbf{D}, \pi_a, \pi_o^*)$ given the navigation task $(\mathbf{S}, \mathbf{D})$ and the agents' policy π_a.

C. Agent-Environment Co-Optimization

We use definitions from individual sub-problems (Sec. II-A and II-B) to formulate the problem of agent-environment co-optimization, which searches for an optimal pair of the navigation policy and the generative model (π_a^*, π_o^*) that maximizes the navigation performance of the multi-agent system, i.e., the objective $f(\mathbf{S}, \mathbf{D}, \pi_a, \pi_o)$, in a joint manner. Specifically, we define the agent-environment co-optimization problem as

$$\begin{aligned} &\underset{\pi_a, \pi_o}{\operatorname{argmax}} \ f(\mathbf{S}, \mathbf{D}, \pi_a, \pi_o) \\ &\text{s.t.} \quad \pi_a(\mathbf{U}_a|\mathbf{X}_a) \in \mathcal{U}_a, \ \pi_o(\mathcal{O}|\mathbf{S}, \mathbf{D}) \in \mathcal{P}_o, \end{aligned} \tag{1}$$

where $\mathcal{U}_a$ is the action space of the agents and $\mathcal{P}_o$ is the feasible set of the obstacle layout. The preceding problem is difficult due to the following challenges:

(i) The objective function $f(\mathbf{S}, \mathbf{D}, \pi_a, \pi_o)$ depends on the multi-agent system and the environment. It is, however, impractical to model this relationship analytically, hence, precluding the application of model-based methods. For the same reason, heuristic methods will perform poorly.
(ii) The navigation policy $\pi_a(\mathbf{U}_a|\mathbf{X}_a)$ generates sequential agent actions based on instantaneous states across time steps, while the generative model $\pi_o(\mathcal{O}|\mathbf{S}, \mathbf{D})$ generates a one-shot (time-invariant) environment configuration based on the navigation task *before* agents start to move. The navigation policy needs unrolling to be evaluated for any generated environment. Hence, π_a and π_o have different inner-working mechanisms and cannot be optimized in the same way, which complicates the joint optimization.
(iii) The navigation policy $\pi_a(\mathbf{U}_a|\mathbf{X}_a)$ and the generative model $\pi_o(\mathcal{O}|\mathbf{S}, \mathbf{D})$ are arbitrary mappings from $\mathbf{X}_a$ and $(\mathbf{S}, \mathbf{D})$ to $\mathbf{U}_a$ and $\mathcal{O}$, which can take any function forms and thus are infinitely dimensional.
(iv) The action space of the agents $\mathcal{U}_a$ can be discrete or continuous and the feasible set $\mathcal{P}_o$ can be convex or non-convex, leading to complex constraints on the feasible solution.

Due to the above challenges, we propose to solve problem (1) with a model-free learning-based approach, in which we alternate between two optimization sub-objectives. Specifically, we parameterize the navigation policy $\pi_a(\mathbf{U}_a|\mathbf{X}_a)$ by a graph

neural network with parameters $\boldsymbol{\theta}_a$, and the generative model $\pi_o(\mathcal{O}|\mathbf{S},\mathbf{D})$ by a deep neural network with parameters $\boldsymbol{\theta}_o$. We update $\boldsymbol{\theta}_a$ with actor-critic reinforcement learning (RL), and $\boldsymbol{\theta}_o$ with a policy gradient ascent (unsupervised learning), in an alternating manner, to find an optimal pair $(\boldsymbol{\theta}_a^*, \boldsymbol{\theta}_o^*)$ that maximizes the navigation performance. The proposed approach overcomes challenge **(i)** with its model-free implementation, challenge **(ii)** with its alternating optimization, challenge **(iii)** with its finite parameterization, and challenge **(iv)** with its appropriate selection of policy / generative distributions.

III. Coordinated Optimization Methodology

We begin by re-formulating problem 1 as an optimization problem with two sub-objectives:

$$\underset{\pi_a}{\operatorname{argmax}} \ f(\mathbf{S},\mathbf{D},\pi_a,\mathcal{E}) \text{ s.t. } \pi_a(\mathbf{U}_a|\mathbf{X}_a)\in\mathcal{U}_a, \tag{2}$$

$$\underset{\pi_o}{\operatorname{argmax}} \ f(\mathbf{S},\mathbf{D},\pi_a,\pi_o) \text{ s.t. } \pi_o(\mathcal{O}|\mathbf{S},\mathbf{D})\in\mathcal{P}_o. \tag{3}$$

Sub-objective (2) optimizes the navigation policy given the environment $\mathcal{E}$, while sub-objective (3) optimizes the generative model given the navigation policy of the agents π_a.

A. Navigation Policy

We are interested in synthesizing a decentralized navigation policy π_a that generates sequential agent actions as a function of local observations. To do so, we define a Markov decision process wherein, at time t, agents observe states $\mathbf{X}_a^{(t)}=[\mathbf{x}_{a1}^{(t)},...,\mathbf{x}_{an}^{(t)}]$ and take actions $\mathbf{U}_a^{(t)}=[\mathbf{u}_{a1}^{(t)},...,\mathbf{u}_{an}^{(t)}]$. Each agent A_i only has access to its own state $\mathbf{x}_{ai}^{(t)}$ (e.g., position and velocity), and communicates with its neighbors to collect $\mathbf{X}_{a\mathcal{N}_i}^{(t)}=\{\mathbf{x}_{aj}^{(t)}\}_{j\in\mathcal{N}_i}$, where $\mathcal{N}_i$ is the neighbor set within communication range. The navigation policy π_a generates the action $\mathbf{u}_{ai}^{(t)}$ based on the local states $\{\mathbf{x}_{ai}^{(t)},\mathbf{X}_{a\mathcal{N}_i}^{(t)}\}$ for $i=1,...,n$, and these actions $\mathbf{U}_a^{(t)}$ control agents to transit from $\mathbf{X}_a^{(t)}$ to $\mathbf{X}_a^{(t+1)}$ based on the probability function $P_a(\mathbf{X}_a^{(t+1)}|\mathbf{X}_a^{(t)},\mathbf{U}_a^{(t)},\mathcal{E})$, which is a distribution of states conditioned on the previous states, actions and environment. Let $r_{ai}(\mathbf{X}_a^{(t)},\mathbf{U}_a^{(t)},\mathcal{E})$ be the reward of agent A_i, representing the instantaneous navigation performance at time t, which depends on agent states $\mathbf{X}_a^{(t)}$, agent actions $\mathbf{U}_a^{(t)}$ and environment configurations $\mathcal{E}$ (e.g., obstacle positions and sizes). The reward of the multi-agent system is the performance averaged over n agents, i.e., $r_a^{(t)}=\sum_{i=1}^n r_{ai}(\mathbf{X}_a^{(t)},\mathbf{U}_a^{(t)},\mathcal{E})/n$. With the discount factor γ that accounts for the future rewards, the expected discounted reward is

$$R_a(\mathbf{S},\mathbf{D},\pi_a,\mathcal{E})=\mathbb{E}\Big[\sum_{t=0}^{\infty}\gamma^t r_a^{(t)}\Big] \tag{4}$$

where the initial states $\mathbf{X}_a^{(0)}$ are determined by the initial and goal positions $(\mathbf{S},\mathbf{D})$, and $\mathbb{E}[\cdot]$ is w.r.t. the navigation policy π_a and the state transition probability P_a. By parameterizing π_a with an information processing architecture $\boldsymbol{\Phi}_a(\mathbf{X}_a,\boldsymbol{\theta}_a)$ of parameters $\boldsymbol{\theta}_a$, we can represent the expected discounted reward as $R_a(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_a,\mathcal{E})$ and the goal is to find the optimal $\boldsymbol{\theta}_a^*$ that maximize $R_a(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_a,\mathcal{E})$. The expected discounted reward in (4) is equivalent to the sub-objective in (2) and we can re-formulate the problem in the RL domain as

$$\underset{\boldsymbol{\theta}_a}{\operatorname{argmax}} \ R_a(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_a,\mathcal{E}) \text{ s.t. } \boldsymbol{\Phi}_a(\mathbf{X}_a,\boldsymbol{\theta}_a)\in\mathcal{U}_a. \tag{5}$$

B. Generative Model

We use a generative model to design the environment. Different from the navigation policy π_a that synthesizes sequential time-varying agent actions, the generative model π_o generates a static environment (i.e., obstacle layout) based on agents' initial and goal positions in one shot before agents start to move. This allows to re-formulate (3) as a stochastic optimization problem. To do so, we assume the initial positions $\mathbf{S}$, goal positions $\mathbf{D}$ and *parameterized* navigation policy $\boldsymbol{\Phi}_a(\mathbf{X}_a,\boldsymbol{\theta}_a)$ given, and thus, can measure the navigation performance with the expected discounted reward $R_a(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_a,\mathcal{E})$ [cf. (5)] in any environment $\mathcal{E}$ generated by the generative model π_o. By parameterizing π_o with an information processing architecture $\boldsymbol{\Phi}_o(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_o)$ of parameters $\boldsymbol{\theta}_o$, we can represent the expected discounted reward as $R_a(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_a,\boldsymbol{\theta}_o)$ because $\boldsymbol{\Phi}_o(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_o)$ determines the obstacle layout of the environment $\mathcal{E}$, and use it as the objective function. The goal is to find the optimal $\boldsymbol{\theta}_o^*$ that maximizes $R_a(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_a,\boldsymbol{\theta}_o)$. The stochastic optimization problem is, hence, formulated in an unsupervised manner as

$$\underset{\boldsymbol{\theta}_o}{\operatorname{argmax}} \ \mathbb{E}\big[R_a(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_a,\boldsymbol{\theta}_o)\big] \text{ s.t. } \boldsymbol{\Phi}_o(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_o)\in\mathcal{P}_o, \tag{6}$$

where the expectation $\mathbb{E}[\cdot]$ is w.r.t. the generative model $\boldsymbol{\Phi}_o(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_o)$ and the navigation policy $\boldsymbol{\Phi}_a(\mathbf{X}_a,\boldsymbol{\theta}_a)$.

Remark 1. *A potential question could be why we do not use reinforcement learning for the sub-problem of environment optimization, as well. The rationale is that in this sub-problem, the states are the navigation task and multi-agent policies given by the system, whereas the action is the obstacle layout produced by the generative model. Since the obstacle layout affects only the navigation performance but not the navigation task or multi-agent policies, the action does not affect the states and there is no transition probability between successive "time" steps. That is, historical information does not influence the current states and there exists no Markov decision process. Therefore, the sub-problem of environment optimization does not fit well into the framework of reinforcement learning. As states at each "time" step are independent (not correlated), it is better modeled as a stochastic optimization problem in an unsupervised manner, where the stochasticity is w.r.t. policy distributions of the generative model and multi-agent policies.*

C. Coordinated Optimization

With these preliminaries, we propose to solve the agent-environment co-optimization problem by coupling the two sub-objectives in an alternating manner. In particular, (5) formulates multi-agent navigation as a reinforcement learning problem, while (6) formulates environment optimization as an unsupervised learning problem. The former optimizes the navigation parameters $\boldsymbol{\theta}_a$ given the environment $\mathcal{E}$, while the latter optimizes the generative parameters $\boldsymbol{\theta}_o$ given the navigation policy $\boldsymbol{\Phi}_a(\mathbf{X}_a,\boldsymbol{\theta}_a)$. We coordinate the optimization of

these sub-objectives to search for an optimal pair $(\boldsymbol{\theta}_a^*, \boldsymbol{\theta}_o^*)$ that maximizes the navigation performance. Specifically, we update parameters iteratively, where each iteration consists of two phases: a multi-agent navigation phase and an environment optimization phase. Details of two phases are as follows.

Synthesizing the navigation policy: At iteration k, let $\boldsymbol{\theta}_a^{(k)}$ be the navigation parameters and $\boldsymbol{\theta}_o^{(k)}$ the generative parameters. Given the initial and goal positions $\mathbf{S}$ and $\mathbf{D}$, the generative model $\boldsymbol{\Phi}_o(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o^{(k)})$ generates the obstacle layout $\mathcal{O}^{(k)}$ and determines the environment $\mathcal{E}^{(k)}$. At time t with the agent states $\mathbf{X}_a^{(k,t)}$, the navigation policy $\boldsymbol{\Phi}_a(\mathbf{X}_a^{(k,t)}, \boldsymbol{\theta}_a^{(k)})$ generates actions $\mathbf{U}_a^{(k,t)}$ that move the agents towards $\mathbf{D}$ with local neighborhood information. These actions lead to the new states $\mathbf{X}_a^{(k,t+1)}$ based on the transition probability $P_a(\mathbf{X}_a^{(k,t+1)}|\mathbf{X}_a^{(k,t)}, \mathbf{U}_a^{(k,t)}, \mathcal{E}^{(k)})$ and receive a reward $r_a^{(k,t)}$.

We follow the actor-critic mechanism and make use of a value function to estimate the expected discounted rewards starting from the agent states $\mathbf{X}^{(k,t)}$ and $\mathbf{X}^{(k,t+1)}$ with the navigation parameters $\boldsymbol{\theta}_a^{(k)}$ in the environment $\mathcal{E}^{(k)}$ [cf. (4)], i.e., $V(\mathbf{X}_a^{(k,t)}, \boldsymbol{\theta}_a^{(k)}, \boldsymbol{\omega}^{(k)}, \mathcal{E}^{(k)})$ and $V(\mathbf{X}_a^{(k,t+1)}, \boldsymbol{\theta}_a^{(k)}, \boldsymbol{\omega}^{(k)}, \mathcal{E}^{(k)})$, where $\boldsymbol{\omega}^{(k)}$ are the parameters of the value function at iteration k. The estimation error can be computed as

$$\begin{aligned}\delta = r_a^{(k,t)} &+ \gamma V(\mathbf{X}_a^{(k,t)}, \boldsymbol{\theta}_a^{(k)}, \boldsymbol{\omega}^{(k)}, \mathcal{E}^{(k)}) \\ &- V(\mathbf{X}_a^{(k,t+1)}, \boldsymbol{\theta}_a^{(k)}, \boldsymbol{\omega}^{(k)}, \mathcal{E}^{(k)}),\end{aligned} \tag{7}$$

which is used to update $\boldsymbol{\omega}^{(k)}$ of the value function. Then, the navigation parameters $\boldsymbol{\theta}_a^{(k)}$ are updated with a policy gradient

$$\boldsymbol{\theta}_{a,1}^{(k)} = \boldsymbol{\theta}_a^{(k)} + \Delta\alpha \nabla_{\boldsymbol{\theta}} \log \pi_a(\mathbf{U}_a^{(k)}|\mathbf{X}_a^{(k)})\delta, \tag{8}$$

where $\Delta\alpha$ is the step-size and $\pi_a(\mathbf{U}_a^{(k)}|\mathbf{X}_a^{(k)})$ is the policy distribution specified by the navigation parameters $\boldsymbol{\theta}_a^{(k)}$. The update in (8) is model-free because it does not require the theoretical model (i.e., analytic expression) of the objective function $R_a(\mathbf{X}^{(k,t)}, \boldsymbol{\theta}_a^{(k)}, \mathcal{E}^{(k)})$, but only the error value δ and the gradient of the policy distribution. The error value δ can be obtained through (7), and the policy distribution is known by design choice. By performing the above update recursively ρ_a times, we get $\boldsymbol{\theta}_{a,1}^{(k)}, \boldsymbol{\theta}_{a,2}^{(k)}, \ldots, \boldsymbol{\theta}_{a,\rho_a}^{(k)}$ for some $\rho_a \in \mathbb{Z}_+$. After ρ_a updates, we transition to the environment optimization phase, with navigation parameters updated by $\boldsymbol{\theta}_a^{(k+1)} := \boldsymbol{\theta}_{a,\rho_a}^{(k)}$.

Learning to generate environments: With the (updated) navigation parameters $\boldsymbol{\theta}_a^{(k+1)}$ and the initial and goal positions $\mathbf{S}$ and $\mathbf{D}$, we can measure the multi-agent navigation performance and leverage the latter to update the generative parameters $\boldsymbol{\theta}_o^{(k)}$. Specifically, the generative model $\boldsymbol{\Phi}_o(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o^{(k)})$ generates the obstacle layout $\mathcal{O}^{(k)}$ based on agents' initial and goal positions $(\mathbf{S}, \mathbf{D})$ to determine the environment $\mathcal{E}^{(k)}$. In this environment, we run the updated navigation policy $\boldsymbol{\Phi}_a(\mathbf{X}_a^{(k)}, \boldsymbol{\theta}_a^{(k+1)})$ to move agents from $\mathbf{S}$ to $\mathbf{D}$. Doing so results in the objective value $R_a^{(k)} := R_a(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_a^{(k+1)}, \boldsymbol{\theta}_o^{(k)})$ representing the navigation performance of $\boldsymbol{\Phi}_a(\mathbf{X}_a^{(k)}, \boldsymbol{\theta}_a^{(k+1)})$ in $\mathcal{E}^{(k)}$, and allows us to update $\boldsymbol{\theta}_o^{(k)}$ with a gradient ascent

$$\boldsymbol{\theta}_{o,1}^{(k)} = \boldsymbol{\theta}_o^{(k)} + \Delta\beta \ \nabla_{\boldsymbol{\theta}_o} \mathbb{E}\big[R_a^{(k)}\big], \tag{9}$$

where $\Delta\beta$ is the step-size, and $\mathbb{E}[\cdot]$ is w.r.t. $\boldsymbol{\Phi}_o(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o^{(k)})$ and $\boldsymbol{\Phi}_a(\mathbf{X}_a^{(k)}, \boldsymbol{\theta}_a^{(k+1)})$. However, (9) requires computing the gradient $\nabla_{\boldsymbol{\theta}_o} \mathbb{E}\big[R_a^{(k)}\big]$, which, in turn, requires a closed-form differentiable model of the objective function $R_a^{(k)}$. The latter is not available as demonstrated in challenge (i) of Sec. II-C.

We overcome this issue by combing the gradient ascent with the policy gradient. In particular, the policy gradient provides a stochastic and model-free approximation for the gradient of policy functions by exploiting the likelihood ratio property [48]. This allows us to estimate the gradient in (9) as

$$\nabla_{\boldsymbol{\theta}_o} \mathbb{E}\big[R_a^{(k)}\big] = \mathbb{E}\big[R_a^{(k)} \nabla_{\boldsymbol{\theta}_o} \log \pi_o(\mathcal{O}^{(k)}|\mathbf{S}, \mathbf{D})\big], \tag{10}$$

where $\pi_o(\mathcal{O}^{(k)}|\mathbf{S}, \mathbf{D})$ is the probability distribution of the obstacle layout specified by the generative parameters $\boldsymbol{\theta}_o^{(k)}$, referred to as the generative distribution. It translates the computation of the function gradient into a multiplication of two terms: (i) the function value $R_a^{(k)}$ and (ii) the gradient of the generative distribution $\nabla_{\boldsymbol{\theta}_o} \log \pi_o(\mathbf{O}^{(k)}|\mathbf{S}, \mathbf{D})$. The former is available with the observed objective value, and the latter can be computed because the generative distribution is known by design choice. Therefore, (10) is model-free, i.e., it can be implemented without any knowledge about the analytic expression of the objective function. The expectation $\mathbb{E}[\cdot]$ can be approximated by unrolling the generative model and the navigation policy $\mathcal{T}$ times and taking the average. We perform this update $\rho_o \in \mathbb{Z}_+$ times to get $\boldsymbol{\theta}_{o,1}^{(k)}, \boldsymbol{\theta}_{o,2}^{(k)}, ..., \boldsymbol{\theta}_{o,\rho_o}^{(k)}$, and the updated environment generative parameters are $\boldsymbol{\theta}_o^{(k+1)} := \boldsymbol{\theta}_{o,\rho_o}^{(k)}$. Algo. 1 gives an overview of our coordinated method, which can be conducted offline.

Algorithm 1 Coordinated Optimization

Input: initial agent positions $\mathbf{S}$, goal positions $\mathbf{D}$, navigation parameters $\boldsymbol{\theta}_a^{(0)}$, generative parameters $\boldsymbol{\theta}_o^{(0)}$
for $k = 0, 1, 2, \ldots$ **do**
(i) Synthesizing the multi-agent navigation policy:
1. Generate the obstacle layout $\mathcal{O}^{(k)}$ with generative model $\boldsymbol{\Phi}_o(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_o^{(k)})$ and determine the environment $\mathcal{E}^{(k)}$
2. Follow the actor-critic mechanism to update navigation parameters $\boldsymbol{\theta}_a^{(k)}$ with policy gradient ρ_a times [cf. (8)]
3. Define new navigation parameters $\boldsymbol{\theta}_a^{(k+1)} = \boldsymbol{\theta}_{a,\rho_a}^{(k)}$
(ii) Learning to generate the environment:
4. Compute the navigation reward $R_a(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_a^{(k+1)}, \mathcal{E}^{(k)})$ with new navigation parameters $\boldsymbol{\theta}_a^{(k+1)}$
5. Compute the policy gradient $\nabla_{\boldsymbol{\theta}_o} \mathbb{E}\big[R_a^{(k)}\big]$ [cf. (10)]
6. Update generative parameters $\boldsymbol{\theta}_o^{(k)}$ with gradient ascent ρ_o times and define new parameters $\boldsymbol{\theta}_o^{(k+1)} = \boldsymbol{\theta}_{o,\rho_o}^{(k)}$
end for

In summary, the coordinated optimization method *couples* multi-agent navigation with environment generation. It optimizes the agents' navigation policy with RL for long-term reward maximization and the generative model with unsupervised learning for one-shot stochastic optimization, representing a new way of combining two machine learning paradigms. The resulting navigation policy and generative model adapt to each other, jointly maximizing the performance. Fig. 2 summarizes this approach. Note that an end-to-end learning without coordination (either RL or unsupervised learning alone) is ill-suited for agent-environment co-optimization because the

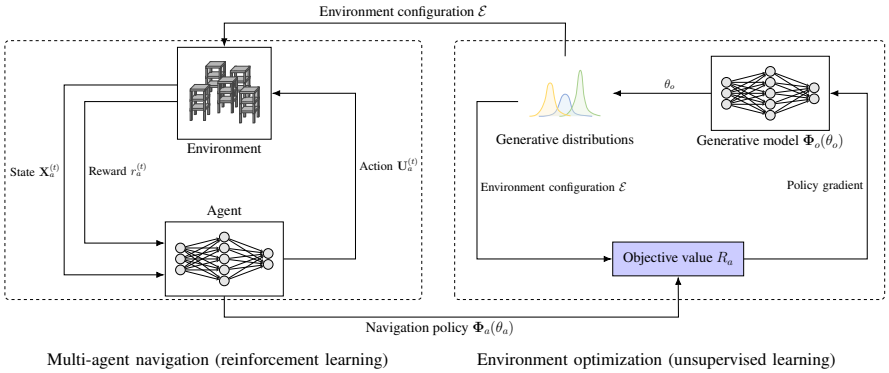


Figure 2. General framework of the agent-environment coordinated optimization method, which alternates between two phases. Given the environment configuration $\mathcal{E}$, the multi-agent navigation phase updates the navigation policy $\mathbf{\Phi}_a(\boldsymbol{\theta}_a)$ through reinforcement learning. Given the navigation policy $\mathbf{\Phi}_a(\boldsymbol{\theta}_a)$, the environment optimization phase updates the generative model $\mathbf{\Phi}_o(\boldsymbol{\theta}_o)$ with policy gradient ascent based on an unsupervised stochastic optimization problem (6). The coordinated optimization method conducts these two phases in an alternating manner, searching for an optimal pair of the environment configuration and the agents' policy.

two models have different inner-working mechanisms, i.e., the navigation policy optimizes sequential actions for a long-term reward [cf. (4)] but the generative model generates a one-shot environment for an instantaneous objective value [cf. (6)]. Moreover, our coordinated scheme is more interpretable and provides a wider range of customization options.

IV. Convergence Analysis

In this section, we analyze the convergence of the proposed method and provide an interpretation for the coordinated optimization scheme by exploring its relation with ordinary differential equations. Specifically, solving the agent-environment co-optimization problem (1) with our method is equivalent to solving a time-varying non-convex optimization problem

$$\min_{\boldsymbol{\theta}_o} \quad g(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_a^{(k)},\boldsymbol{\theta}_o) \quad \text{s.t.} \quad \mathbf{\Phi}_o(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_o)\in\mathcal{P}_o \qquad (11)$$

with policy gradient ascent of step-size $\Delta\beta$, where $g(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_a^{(k)},\boldsymbol{\theta}_o)=-f(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_a^{(k)},\boldsymbol{\theta}_o)$ is the inverse objective function, the navigation policy π_a and the generative model π_o are represented by their parameters $\boldsymbol{\theta}_a^{(k)}$ and $\boldsymbol{\theta}_o$, and $\boldsymbol{\theta}_a^{(k)}$ are updated by the actor-critic mechanism with step-size $\Delta\alpha$ [cf. (8)]. Problem (11) considers the co-optimization problem from the perspective of environment optimization and regards the navigation policy as time-varying parameters that modify its optimization objective across iterations k. The problem is typically non-convex because of the unknown function g and the nonlinear parameterization $\mathbf{\Phi}_o$.

The time-varying problem (11) changes with the navigation parameters $\boldsymbol{\theta}_a$, which motivates the definition of a "time" variable α to capture the inherent variation. In particular, $\boldsymbol{\theta}_a$ is updated by the actor-critic mechanism with policy gradient [cf. (8)]. By assuming the policy gradient is bounded, the variation of $\boldsymbol{\theta}_a$ is determined by the step-size $\Delta\alpha$. In this context, we can represent $\boldsymbol{\theta}_a^{(k)}$ as a function of the initial $\boldsymbol{\theta}_a^{(0)}$, the step-size $\Delta\alpha$ and the number of iterations k, i.e., $\boldsymbol{\theta}_a^{(k)}=\Theta\big(\boldsymbol{\theta}_a^{(0)},\alpha^{(k)}\big)$ where $\alpha^{(k)}=k\Delta\alpha$ is the step length. Define a continuous "time" variable α, which is instantiated as $\alpha^{(k)}$ at iteration k. By combining this definition with problem (11), we can formulate a continuous time-varying problem as

$$\min_{\boldsymbol{\theta}_o} \quad g(\mathbf{S},\mathbf{D},\alpha,\boldsymbol{\theta}_o) \quad \text{s.t.} \quad \mathbf{\Phi}_o(\mathbf{S},\mathbf{D},\boldsymbol{\theta}_o)\in\mathcal{P}_o, \qquad (12)$$

where $\alpha\geq 0$ denotes the scale of "time" and $\boldsymbol{\theta}_o$ are the decision variables. Problem (12) is the continuous limit of (11), where the former reduces to the latter if particularizing α to discrete values $\{\alpha^{(k)}\}_k$ corresponding to different iterations. Because of the non-convexity, we consider the first-order stationary condition as the metric for convergence analysis and define the local minimum trajectory of problem (12).

Definition 1 (Local minimum trajectory)**.** A p-dimensional continuous trajectory $\mathcal{T}(\alpha):[0,+\infty]\to\mathbb{R}^p$ is a local minimum trajectory of the non-convex time-varying optimization problem (12) if each point of $\mathcal{T}(\alpha)$ is a local minimum of problem (12) at "time" $\alpha\in[0,+\infty)$.

The local minimum trajectory $\mathcal{T}(\alpha)$ tracks the sequence of local solutions for the continuous non-convex time-varying optimization problem (12).[1] If freezing the "time" variable α at a particular value, problem (12) becomes a classic (time-invariant) non-convex optimization problem and the local minimum trajectory becomes a stationary local solution.

Assumption 1. The inverse objective function $g(\mathbf{S},\mathbf{D},\alpha,\boldsymbol{\theta}_o)$ of problem (12) is differentiable w.r.t. α and $\boldsymbol{\theta}_o$, and the gradient is Lipschitz continuous w.r.t. $\boldsymbol{\theta}_o$, i.e.,

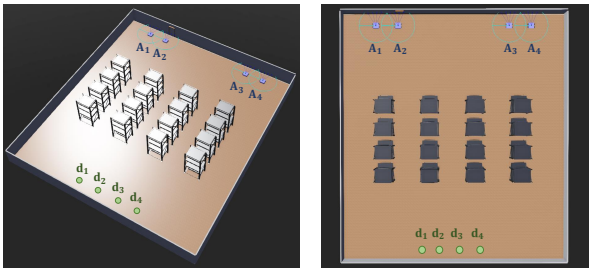
$$\|\nabla_{\boldsymbol{\theta}_o} g(\mathbf{S},\mathbf{D},\alpha,\boldsymbol{\theta}_{o,1})-\nabla_{\boldsymbol{\theta}_o} g(\mathbf{S},\mathbf{D},\alpha,\boldsymbol{\theta}_{o,2})\|\leq C_L\|\boldsymbol{\theta}_{o,1}-\boldsymbol{\theta}_{o,2}\|$$

for any $\boldsymbol{\theta}_{o,1},\boldsymbol{\theta}_{o,2}\in\mathbb{R}^p$, where $\|\cdot\|$ is the vector norm, C_L is the Lipschitz constant, and p is the dimension of $\boldsymbol{\theta}_o$.

Assumption 2. The policy gradient in (10) estimates the true gradient in (9) within an error neighborhood of size ε.

Assumption 1 indicates that the gradient of the inverse objective function $\nabla_{\boldsymbol{\theta}_o} g(\mathbf{S},\mathbf{D},\alpha,\boldsymbol{\theta}_o)$ does not change faster than linear w.r.t. the generative parameters $\boldsymbol{\theta}_o$, which is a standard assumption in optimization theory [49]. Here, the Lipschitz continuity is assumed w.r.t. the objective function rather than the designed reward. The objective function is an analytic expression characterizing the relationship between the environment, agents and performance, which is challenging to model due to their unclear relationship, while the designed

[1]The local minimum trajectory is a sequence of local minimum solutions that lie in the model parameter space, not the robot physical space.



(a) Side view **(b)** Top-down view

Figure 3. Environment scenario with 4 agents and 4 obstacles in a warehouse setting. The agents $\{A_i\}_{i=1}^4$ represent robots, the obstacles represent rectangular shelves of size 1×4, and the green circles $\{\mathbf{d}_i\}_{i=1}^4$ represent the goal positions. (a) Side view of the environment. (b) Top-down view of the environment.

reward approximates the value of the objective function and leverages the latter value to estimate the gradient of the objective function for optimization. Assumption 2 implies that the policy gradient approximates the true gradient within an error neighborhood ε. The value of ε depends on the performance of the policy gradient, which has exhibited success in a wide array of reinforcement learning problems [48], [50].

In real-world applications, it is neither practical nor realistic to have solutions that abruptly change over "time". It is standard to impose a soft constraint that penalizes the inverse objective function with the deviation of its solution from the one obtained at the previous "time" [51]. The resulting discrete time-varying optimization problem with proximal regularization (except for the initial problem) becomes

$$\min_{\boldsymbol{\theta}_o} \; g(\mathbf{S}, \mathbf{D}, \alpha^{(0)}, \boldsymbol{\theta}_o) \tag{13}$$

$$\min_{\boldsymbol{\theta}_o} \; g(\mathbf{S},\mathbf{D},\alpha^{(k)},\boldsymbol{\theta}_o)+\frac{\eta\|\boldsymbol{\theta}_o-\boldsymbol{\theta}_o^{(k-1)*}\|}{2\Delta\alpha}, \text{for } k=1,2,\ldots \tag{14}$$

where $\boldsymbol{\theta}_o^{(k-1)*}$ is a local minimum of problem (14) at iteration $k-1$ and η is a regularization parameter that weighs the regularization term to the inverse objective function. The first term in (14) is the inverse objective, while the second term in (14) denotes the deviation of the decision variable $\boldsymbol{\theta}_o$ at current iteration k from the local minimum at previous iteration $k-1$. By taking the derivative and using the first-order stationary condition, the local minimum of (14) at iteration k satisfies

$$\nabla_{\boldsymbol{\theta}_o} g(\mathbf{S}, \mathbf{D}, \alpha^{(k)}, \boldsymbol{\theta}_o^{(k)*})+\eta\frac{\boldsymbol{\theta}_o^{(k)*}-\boldsymbol{\theta}_o^{(k-1)*}}{\Delta\alpha}=0, \tag{15}$$

where the first term represents the inverse objective gradient and the second term represents the change of the local minimum at two successive iterations. When the step-size approaches zero $\Delta\alpha \to 0$, the continuous limit of (15) becomes the ordinary differential equation (ODE) as

$$\eta\boldsymbol{\theta}_o'(\alpha)=-\nabla_{\boldsymbol{\theta}_o} g(\mathbf{S}, \mathbf{D}, \alpha, \boldsymbol{\theta}_o(\alpha)), \;\; \boldsymbol{\theta}_o(0)=\boldsymbol{\theta}_o^{(0)*}, \tag{16}$$

where $\boldsymbol{\theta}_o(\alpha)$ is a dynamic function of α and $\boldsymbol{\theta}_o'(\alpha)$ is the derivative of $\boldsymbol{\theta}_o(\alpha)$ w.r.t. α. The solution of the ODE (16) is a local minimum trajectory of the continuous time-varying optimization problem with proximal regularization [cf. (13)-(14) with $\Delta\alpha \to 0$]. When $\eta = 0$, the ODE (16) reduces to the first-order stationary condition of the time-varying optimization problem (12). Given the initial point $\boldsymbol{\theta}_o(0)$, we assume there exists a local minimum trajectory $\mathcal{T}(\alpha)$ [Def. 1] satisfying that $\boldsymbol{\theta}_o(\alpha) - \mathcal{T}(\alpha)$ lies in a compact set for $\alpha \geq 0$. With these preliminaries, we formally show the convergence of the proposed coordinated optimization method.

Theorem 1. *Consider the non-convex time-varying optimization problem* (12) *satisfying Assumption 1 w.r.t.* C_L *and the policy gradient in* (9) *satisfying Assumption 2 w.r.t.* ε. *Let* $\boldsymbol{\theta}_o(\alpha)$ *be a solution of the ODE* (16) *with the regularization parameter* η *and* $\{\boldsymbol{\theta}_o^{(k)}\}_k$ *be the discrete sequence of the generative parameters updated by the proposed method, where the step-size of the policy gradient ascent is* $\Delta\beta = \Delta\alpha/\eta$ *[cf.* (9)*]. Let also the initial* $\boldsymbol{\theta}_o^{(0)} = \boldsymbol{\theta}_o(0)$ *be a local minimum of* $g(\mathbf{S}, \mathbf{D}, 0, \boldsymbol{\theta}_o)$. *Then, for any* $\epsilon > 0$ *and "time" horizon* T, *there exists a step-size* $\Delta\alpha$ *such that*

$$\|\boldsymbol{\theta}_o^{(k)} - \boldsymbol{\theta}_o(\alpha^{(k)})\| \leq \epsilon + C\varepsilon \tag{17}$$

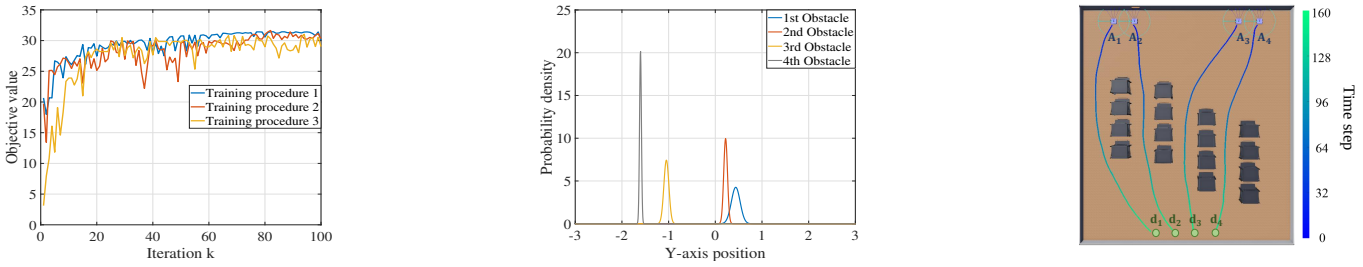
for all iterations $k = 1, \ldots, \lfloor T/\Delta\alpha \rfloor$ *within the horizon, where* C *is a constant depending on* C_L, η *and* T *[cf.* (29)*].*

Proof. See Appendix A. □

Theorem 1 states that the proposed co-optimization method converges to the local minimum trajectory of a time-varying non-convex optimization problem, i.e., the solution of the ODE (16), up to a limiting error neighborhood. The latter consists of two additive terms: *i)* the first term ϵ is the desired error, which can be arbitrarily small; *ii)* the second term is proportional to the accuracy of the policy gradient ε, which becomes null as the policy gradient approaches the true gradient. The update step-sizes of the navigation policy $\Delta\alpha$ and the generative model $\Delta\beta = \Delta\alpha/\eta$ depend on the Lipschitz constant C_L, the regularization parameter η, the desired error ϵ and the "time" horizon T [cf. (29)]. This indicates that a more accurate solution, i.e., a smaller ϵ, requires a smaller step-size $\Delta\alpha$, which may slow down the training procedure. We note that the total number of iterations $\lfloor T/\Delta\alpha \rfloor$ is inversely proportional to the step size $\Delta\alpha$, which increases as $\Delta\alpha$ decreases and keeps the "time" horizon T unchanged; hence, Theorem 1 holds uniformly in any given "time" horizon $\alpha^{(k)} \in [0, T]$. This result characterizes the convergence behavior of the proposed method and interprets its inherent working mechanism, i.e., performing the proposed method is equivalent to tracking the local minimum trajectory of an associated time-varying non-convex optimization problem.

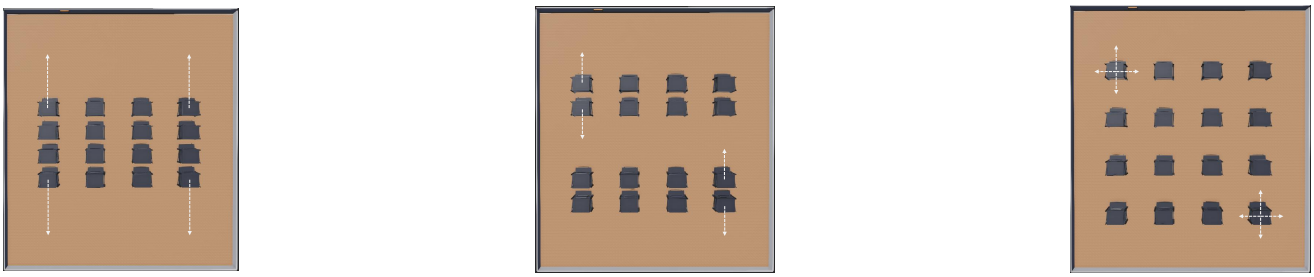
V. Experiments

In this section, we evaluate the coordinated optimization method. First, we conduct a proof of concept to demonstrate its effectiveness. Then, we show how our approach improves the navigation performance in a warehouse setting. Lastly, we characterize the role of the environment in the multi-agent system through our method and show that an appropriate obstacle layout provides "positive" guidance for agents, beyond "negative" obstruction as in conventional beliefs. This finding has not been explored thus far, highlighting the importance of building a symbiotic agent-environment co-evolved system.



(a) **(b)** **(c)**

Figure 4. Proof of concept for agent-environment coordinated optimization. (a) Training procedure of coordinated optimization method. (b) Generative distributions for 4 obstacles. (c) Obstacle layout produced by the generative model and agent trajectories controlled by the navigation policy. Blue-to-green lines are trajectories of agents and the color bar represents the time scale.



(a) Environment I **(b)** Environment II **(c)** Environment III

Figure 5. Three environments in Section V-C. (a) Environment I with 4 obstacles of size 1×4. (b) Environment II with 8 obstacles of size 1×2. (c) Environment III with 16 obstacles of size 1×1.

All experiments are performed on a computer with an Intel Core i7-11700 CPU and a GeForce RTX 3060 GPU.[2]

A. Implementation Details

The agents and obstacles are modeled by their positions $\{\mathbf{p}_{ai}\}_{i=1}^{n}$, $\{\mathbf{p}_{oj}\}_{j=1}^{m}$ and velocities $\{\mathbf{v}_{ai}\}_{i=1}^{n}$. For multi-agent navigation, each agent observes its own state (e.g., position and velocity), communicates with neighboring agents, and generates the desired velocity with received neighborhood information. We integrate an acceleration-constrained mechanism for position changes and an episode ends if all agents reach destinations or episode times out. The maximal acceleration is $1m/s^2$, the maximal velocity is $1.5m/s$, the communication radius is $2m$, the maximal time is 500 steps and each time step is $0.05s$. The goal is to make agents reach destinations quickly while avoiding collision. We define the reward function as

$$r_{ai}^{(t)} = \left(\frac{\mathbf{p}_{ai}^{(t)} - \mathbf{d}_i}{\|\mathbf{p}_{ai}^{(t)} - \mathbf{d}_i\|_2} \cdot \frac{\mathbf{v}_{ai}^{(t)}}{\|\mathbf{v}_{ai}^{(t)}\|_2} \right) \|\mathbf{v}_{ai}^{(t)}\|_2 + C_p^{(t)} \text{ at time step } t.$$

The first term rewards fast movement to the goal position at the desired orientation, and the second term represents the collision penalty with $C_p^{(t)} = -C_p$ if there exists a collision and $C_p^{(t)} = 0$ otherwise, where C_p is a positive constant. We parameterize the navigation policy with a single-layer GNN, where message aggregation and feature update functions are multi-layer perceptrons. We train the policy using PPO [52].

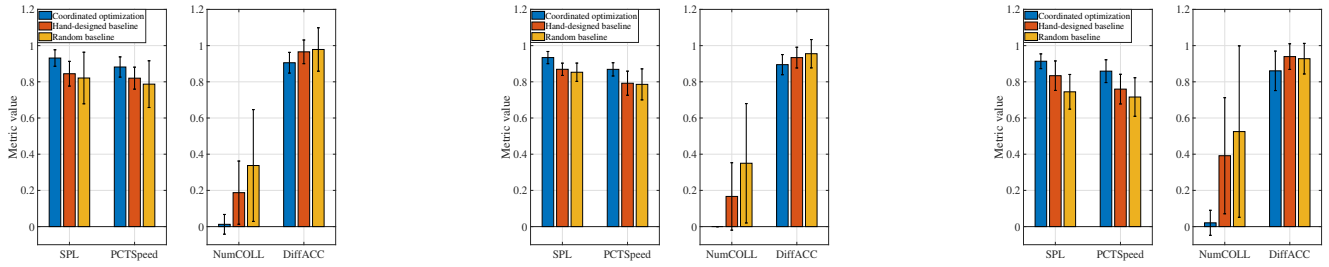
The environment generative model produces m generative distributions, each controlling the configuration (e.g., position and size) of one obstacle. We consider generative distributions as truncated Gaussian distributions – see Appendix B, where the dimension, lower and upper bounds depend on specific environment settings. For example in Fig. 3a, each obstacle is a shelf in the warehouse. The obstacle position on the x-axis is fixed and that on the y-axis can be adjusted in $[-4, 4]$ by sliding along a linear track. The generative distribution is a 1-D truncated Gaussian distribution with a lower bound -4 and an upper bound 4. We parameterize the generative model with a 3-layer DNN of $(32, 64, 32)$ units and ReLU nonlinearity.

We employ Webots as our 3D simulator, which contains realistic physical properties to simulate real-world conditions. Specifically, we design robots with joints, motors, communication modules and omnidirectional wheels to consider realistic physical constraints. We follow [53] to design materials of the floor and wheels, and tune the Coulomb friction coefficient to define contact properties for wheel movements. Moreover, we configure motor constraints, such as maximum angular velocity and torque for each wheel, to mimic real-world motor behaviors. All objects in Webots, including robot bodies, wheels, sensors and obstacles, have inertia, and the collision detection is based on the mesh geometry imposed on entities within the simulator. Each robot has a receiver and an emitter to broadcast and receive the state from its neighboring robots via communication. The low-level control is handled by model predictive control, which converts the navigation policy output to each wheel's speed using inverse kinematics. Therefore, our simulations account for realistic physical constraints instead of considering robots as simple point masses.

B. Proof of Concept

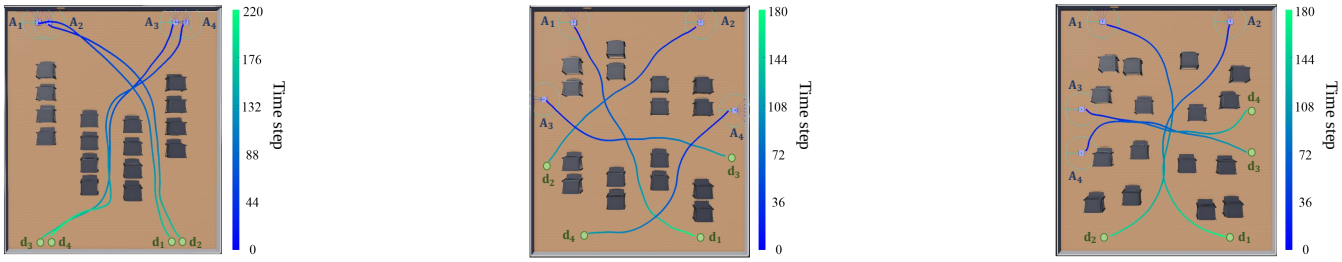
First, we conduct a proof of concept with 4 agents and 4 obstacles. We consider an environment of size 10×12 in the

[2]Parallel computation with more CPUs and GPUs can speed up our method, depending on specific time requirements in practical applications.



(a) **(b)** **(c)**

Figure 6. Performance with expectation and standard deviation of agent-environment coordinated optimization method compared with two baselines in three environments. A higher value of SPL or PCTSpeed represents a better performance, while a lower value of NumCOLL or DiffACC represents better safety or more comfort. (a) Environment I. (b) Environment II. (c) Environment III.



(a) Optimized Environment I **(b)** Optimized Environment II **(c)** Optimized Environment III

Figure 7. Examples of agent-environment co-optimization in three environments. Blue robots $\{A_i\}_{i=1}^4$ represent agents at initial positions, green circles $\{\mathbf{d}_i\}_{i=1}^4$ represent goal positions, and black boxes represent obstacles (shelves). Colored lines from blue to green are agent trajectories and the color bar is the time scale. (a) Optimized Environment I. (b) Optimized Environment II. (c) Optimized Environment III.

warehouse setting as depicted in Fig. 3. The agents are robots of size 0.4, and the obstacles are rectangular shelves of size 1×4. The obstacle positions on the x-axis are fixed, yet are configurable on the y-axis in the range of $[-4, 4]$ to improve the navigation performance.

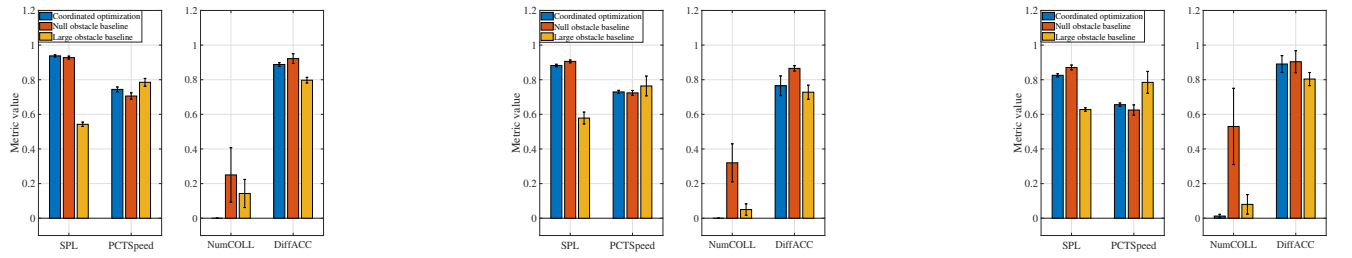
Fig. 4a shows that the objective value increases with the number of iterations and approaches a stationary condition, which corroborates the convergence analysis in Theorem 1. Fig. 4b plots the generative distributions for four obstacles, respectively, which demonstrate how the generative model places the obstacles in the environment. First, the standard deviation of the distribution is small, i.e., the probability density concentrates in a small region. This indicates that there is little variation in the environment generation (hence, the resulting performance). Second, the generative model produces an irregular obstacle layout that differs from the hand-designed one, which implies that structure irregularity is capable of providing navigation help for the multi-agent system. Fig. 4c shows the agent trajectories controlled by our navigation policy in the generated environment, where agents move smoothly from initial positions to their goals without collision.

C. Performance Evaluation

We further evaluate our approach in three different cluttered environments that resemble a warehouse setting:

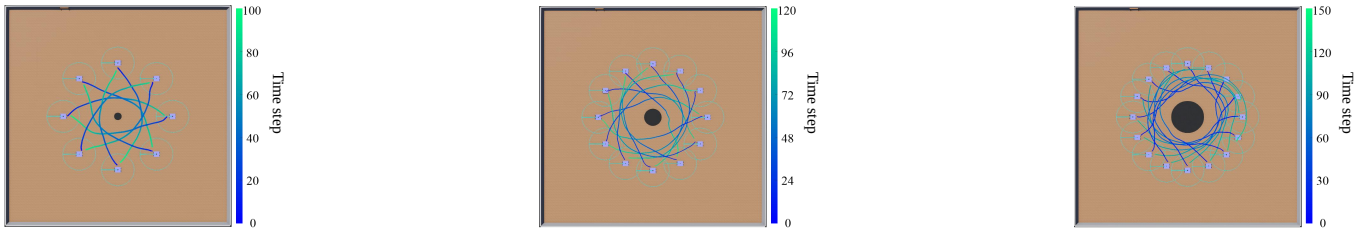
Environment I is equipped with 4 obstacles of size 1×4 – see Fig. 5a. The obstacle positions on the x-axis are fixed, and those on the y-axis can be re-configured in the range $[-4, 4]$.
Environment II is equipped with 8 obstacles of size 1×2 – see Fig. 5b. The obstacle positions on the x-axis are fixed, and those on the y-axis can be re-configured in $[-4, 0]$ or $[0, 4]$.
Environment III is equipped with 16 obstacles of size 1×1 – see Fig. 5c. The obstacle positions can be re-configured along both x- and y-axes in a neighborhood of size 2×2.
We systematically consider Environments I-III with different degrees of flexibility w.r.t. obstacle positions, corresponding to different practical constraints. For example, Environment I re-arranges obstacles along the y-axis assuming there exist sliding tracks on the y-axis, while Environment III re-arranges obstacles along both x- and y-axes assuming they are flexible in any direction.

We measure the navigation performance with four metrics: *(i)* Success weighted by Path Length (SPL) [54], *(ii)* the percentage to the maximal speed (PCTSpeed), *(iii)* the average number of "collisions" (NumCOLL), and *(iv)* the average finite difference of the acceleration (DiffACC). SPL is a stringent measure that combines success rate with path length overhead, and has been widely used as a primary quantifier in comparing navigation performance [54]–[56]. PCTSpeed represents the ratio of the average speed to the maximal one, and provides complementary information regarding the moving speed along trajectories. These two metrics are normalized to $[0, 1]$ with a higher value representing a better performance, providing a unified exposition. NumCOLL counts the number of 'collisions' averaged over agents, where an agent is considered 'in collision' at time t if it is within the safety range of another agent or obstacle, and measures the safety of the multi-agent system. DiffACC represents the change of agents' acceleration averaged over time steps, and indicates the comfort (smoothness) of the navigation procedure [57], [58]. For the latter two metrics, a lower value represents a better performance, i.e., better safety and more comfort – see Appendix C. Our results



(a) **(b)** **(c)**

Figure 8. Performance of agent-environment coordinated optimization compared with two baseline scenarios in the circular setting with different numbers of agents. A higher value of SPL or PCTSpeed represents a better performance, while a lower value of NumCOLL or DiffACC represents better safety or more comfort. (a) $n = 8$ agents. (b) $n = 12$ agents. (c) $n = 16$ agents.



(a) Environment with 8 agents **(b)** Environment with 12 agents **(c)** Environment with 16 agents

Figure 9. Examples of agent-environment co-optimization in the circular setting with different numbers of agents. Blue robots $\{A_i\}_{i=1}^n$ represent agents at initial positions, green circles $\{\mathbf{d}_i\}_{i=1}^n$ represent goals, and the solid black circle represents the obstacle. Colored lines from blue to green are agent trajectories and the color bar represents the time scale. (a) $n = 8$ agents. (b) $n = 12$ agents. (c) $n = 16$ agents.

are averaged over 30 random tasks with 10 random trials for a total of 300 runs. The initial and goal positions of each task are randomly sampled in the environment boundaries, while maintaining a minimum distance between the starting and goal positions of each agent for a non-trivial navigation task.

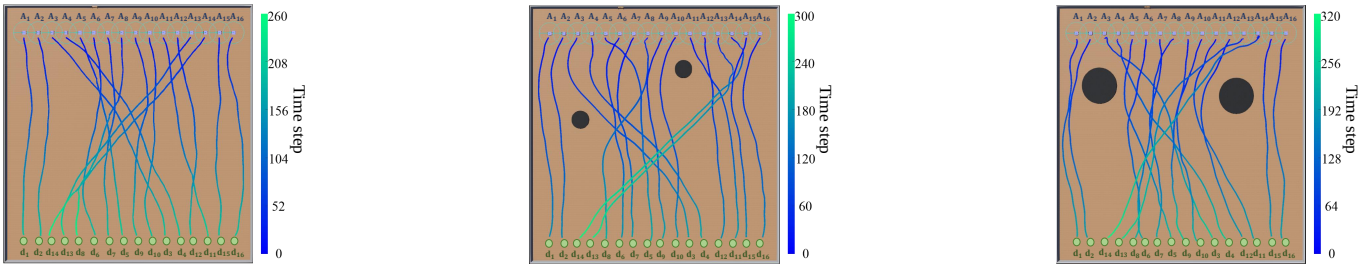
Fig. 6 shows the performance with expectation and standard deviation of the coordinated optimization method. We consider two baselines: a hand-designed environment as shown in Fig. 5 and the randomly generated environment. The hand-designed baseline is a standard environment with a regular obstacle layout, as is common practice. Our method outperforms the baselines consistently with the highest SPL and PCTSpeed and the lowest NumCOLL and DiffACC, and maintains a good performance across different environments. We attribute this behavior to that the generative model adapts the obstacle layout to alleviate environment restrictions, based on which the navigation policy tunes the agent trajectories to improve the performance. The performance improvement is emphasized as the environment structure becomes more complex from Environment I with 4 obstacles to Environment III with 16 obstacles. This is because a greater number of obstacles leads to increasingly challenging navigation scenarios, in which agents derive more benefits from environments that are more compatible with their navigation policies.

Fig. 7 displays examples of the agent-environment co-optimization in Environments I-III. We see that: *(i)* The generative model produces different obstacle layouts depending on different navigation tasks, i.e., different starting and goal positions of agents. The resulting environment exhibits an irregular structure different from the hand-designed one, which facilitates the multi-agent navigation. *(ii)* The generated environment not only creates obstacle-free pathways for agents, but also prioritizes these pathways through structure irregularity and non-symmetry to implicitly de-conflict agents via prioritization. For example in Fig. 7c, while the navigation tasks of A_1 and A_2 are symmetric, the generative model produces an irregular obstacle layout that creates non-symmetric obstacle-free pathways for A_1 and A_2. This prioritizes the two agents and avoids their potential confliction by allowing A_2 to pass the passage before A_1. *(iii)* The navigation policy is compatible with the generative model and selects the optimal trajectories in the generated environment, moving the agents smoothly towards goals. *(iv)* The navigation policy strives to move agents as a team (or sub-team) to remain connected, because the agents rely on their neighborhood information to make collectively meaningful decisions.

These results indicate that our approach establishes a symbiotic agent-environment system, where the environment and the agents rely on and adapt to each other; ultimately, yielding an improved navigation performance.

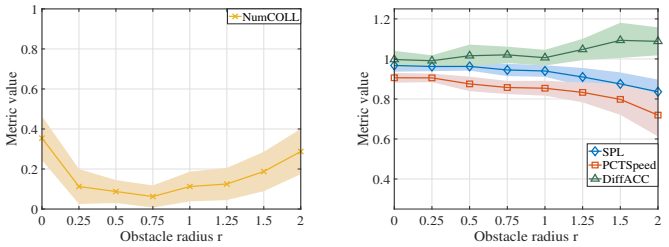
D. Hindrance vs. Guidance

Next, we investigate the role played by the environment in multi-agent navigation by leveraging our coordinated optimization method. Obstacles are typically viewed as "negative" elements that create inaccessible areas or obstruct agent pathways (and hinder ideal trajectories). However, we hypothesize that an appropriate obstacle layout can have "positive" effects by forming designated collision-free zones for the agents to follow. Such an arrangement can provide navigation guidance and help de-conflict agents, especially in dense settings. It is somewhat counterintuitive that it outperforms the empty space. The following experiments aim to verify this hypothesis and to capture the relationship between the environment and agents.



(a) **(b)** **(c)**

Figure 10. Examples of agent-environment co-optimization in Case I of the randomized setting with different obstacle radii. Blue robots $\{A_i\}_{i=1}^{16}$ represent agents at initial positions, green circles $\{\mathbf{d}_i\}_{i=1}^{16}$ represent goals, and black circles represent obstacles. Colored lines from blue to green are agent trajectories and the color bar represents the time scale. (a) Null radius $r = 0$. (b) Radius $r = 0.5$. (c) Radius $r = 1$.



(a) **(b)**

Figure 11. Performance of agent-environment co-optimization in Case I of the randomized setting with different obstacle radii. A higher SPL / PCTSpeed represents a better performance, while a lower NumCOLL / DiffACC represents better safety or more comfort.

That is, we aim to uncover an implicit trade-off between hindrance and guidance roles that the environment may take on.

We consider two environment settings motivated by traffic system design for safe navigation.

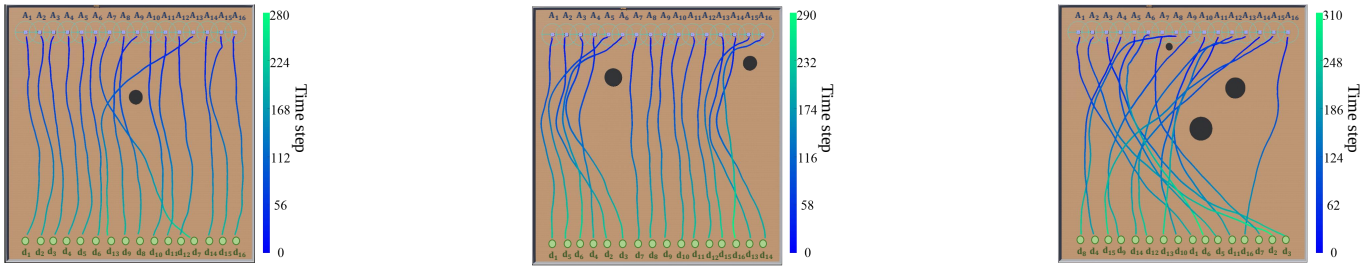
Circular setting. The environment is of size 8×8, where agents are initialized along a circle – see Fig. 9. The goal of the navigation policy is to cross-navigate the agents towards the opposite side while avoiding collisions, and the goal of the generative model is to determine the radius of the circular obstacle at the origin $(0, 0)$ to de-conflict the agents. A large obstacle reduces the reachable space for the agents and may degrade the navigation performance, while a small obstacle provides a negligible guiding effect and may not prevent agent conflicts (e.g., all agents attempt to move along the shortest but intersecting trajectories). We explore this hindrance-guidance trade-off by performing the agent-environment co-optimization to find the optimal obstacle radius.

Fig. 8 shows the performance of the coordinated optimization method with different numbers of agents $n \in \{8, 12, 16\}$. We consider two baseline scenarios: a null obstacle of radius 0 and a large obstacle of radius 2. The presence of the obstacle with an appropriate radius improves the agents' performance with a significantly lower NumCOLL. This improvement scales with the number of agents, which can be explained by that a more cluttered system with a higher density of agents necessitates more guidance from the obstacle to facilitate agent de-confliction. The baseline scenario with a null obstacle, i.e., the empty space, achieves a comparable SPL / PCTSpeed but increases NumCOLL, despite no hindrance along agent trajectories. This is because the agents have to coordinate their navigation trajectories with only neighbors' states, and receive no guidance from the obstacle. Its NumCOLL increases from the scenario with 8 agents to the one with 16 agents. We attribute this behavior to that while inter-agent interactions may be sufficient for de-confliction in a sparse environment with a small number of agents, it is challenging for agents to de-conflict each other in a cluttered environment with a large number of agents and the latter requires guidance from the obstacle. The baseline scenario with a large obstacle results in a lower SPL and a slightly higher NumCOLL because it reduces the reachable space and imposes more restrictions on the feasible solution. The latter blocks the shortest pathways and forces the agents to move along inefficient trajectories.

Fig. 9 plots the obstacle radius of the generative model and agent trajectories of the navigation policy. The obstacle radius generated by our method increases with the number of agents n. For a small n, the agents are sparsely distributed and need less guidance from the obstacle for de-confliction, leading to a small radius that does not hinder agent pathways. For a large n, the agents are densely distributed and require more help from the obstacle for coordination, leading to a large radius that guides the agents moving clockwise towards goals.

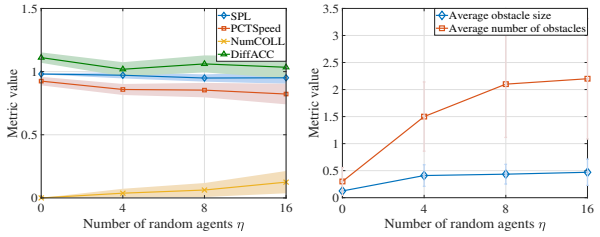
Randomized setting. The environment is of size 18×18 with 16 starting positions $\{\mathbf{s}_i\}_{i=1}^{16}$ on the top and 16 goals $\{\mathbf{d}_i\}_{i=1}^{16}$ on the bottom – see Fig. 10. There are 16 agents $\{A_i\}_{i=1}^{16}$ initialized at $\{\mathbf{s}_i\}_{i=1}^{16}$ and tasked towards $\{\mathbf{d}_i\}_{i=1}^{16}$ in order, i.e., agent A_i is initialized at $\mathbf{s}_i$ and tasked towards $\mathbf{d}_i$ for $i = 1, ..., 16$. The goal of the navigation policy is to move the agents to their goals while avoiding collision among each other, and the goal of the generative model is to determine the obstacle layout between starting and goal positions to facilitate safe navigation. An environment with a large number of obstacles or a large obstacle size hinders agent pathways, while an empty environment might not provide enough guidance for effective agent de-confliction. By leveraging the agent-environment co-optimization, we explore this trade-off to find the optimal obstacle layout for different navigation cases.

Case I. We consider an environment with 2 circular obstacles, where the obstacle radius varies in $[0, 2]$. We randomly switch the initial and goal positions of 8 agents, while keeping the other 8 agents unchanged – see Fig. 10a. The generative model determines the optimal obstacle positions that de-conflict the agents while avoiding hindering their pathways, striking a balance between the guidance and the hindrance roles.



(a) **(b)** **(c)**

Figure 12. Examples of agent-environment co-optimization in Case II of the randomized setting with different numbers of random agents η. Blue robots $\{A_i\}_{i=1}^{16}$ represent agents at initial positions, green circles $\{\mathbf{d}_i\}_{i=1}^{16}$ represent goals, and black circles represent obstacles. Colored lines from blue to green are agent trajectories and the color bar represents the time scale. (a) $\eta = 4$. (b) $\eta = 8$. (c) $\eta = 16$.



(a) **(b)**

Figure 13. (a) Performance of agent-environment coordinated optimization in Case II of the randomized setting with different numbers of random agents η. A higher value of SPL or PCTSpeed represents a better performance, while a lower value of NumCOLL or DiffACC represents better safety or more comfort. (b) Average obstacle radius and average number of obstacles determined by coordinated optimization method with different numbers of random agents η.

Fig. 11a shows that NumCOLL first decreases with the obstacle radius r, and then increases for large r. The former is because an initial increase in r allows the presence of obstacles in an empty environment, which provides the operation space for our method to de-conflict the agents with an appropriate obstacle layout. The result for large r corresponds to the fact that further increasing r occupies more collision-free space but provides little additional guidance, leading to an increased NumCOLL. Fig. 11b shows that SPL, PCTSpeed decrease and DiffACC increases slightly with r, while all maintaining a good performance. This slight degradation is because *(i)* increasing r reduces the reachable space in the environment and *(ii)* some agents make a detour under the obstacle guidance to avoid collision with the other agents.

Fig. 10 plots examples of how our method optimizes obstacle positions with different obstacle radii. We see that different obstacle positions and sizes have different impact on agents' trajectories, and the obstacles are typically placed around the intersections of agents' trajectories to separate the crowd and de-conflict the agents. For example, with no obstacle in Fig. 10a, A_{12}, A_{13}, A_{14} all attempt to move fast along short paths, which increases the risk of agent collisions. With the presence of obstacles in Fig. 10c, A_{13}, A_{14} are guided to slow down and make a detour for A_{12} to avoid potential collisions.

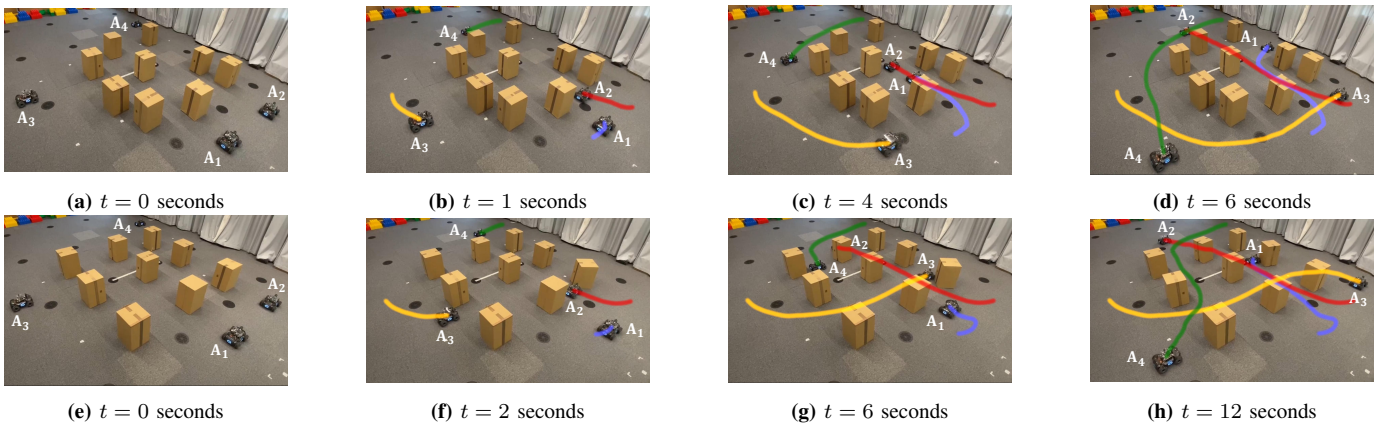
Case II. We consider an environment with at most 4 circular obstacles. We randomly switch the initial and goal positions of η agents, where $\eta \in [0, 16]$ represents the randomness of the navigation task, while keeping the rest $16-\eta$ agents unchanged – see Fig. 12. The generative model determines: *(i)* the number of obstacles presented in the environment; *(ii)* the obstacle radii and *(iii)* the obstacle positions, to de-conflict the agents while avoiding blocking their pathways.

Fig. 13a shows the performance with different numbers of random agents η. The proposed method maintains a good performance across varying agent randomness, highlighting its capacity in handling different navigation scenarios. SPL and PCTSpeed decrease and NumCOLL increases slightly with the agent randomness because the navigation task becomes more challenging as the initial and goal positions of the agents get more random. Fig. 13b plots the average obstacle size and the average number of presented obstacles with different η. Both increase with the navigation randomness, while the increasing rates decrease as η becomes large. The former is because when the randomness is small, the agents are capable of handling the de-confliction and need little help from the obstacles; when the randomness increases and the system becomes cluttered, the agents need more or larger obstacles to provide guidance for de-confliction. The latter is because as the obstacle number or radius reaches a saturated level, it has sufficient capacity to guide the agents and thus, reduces the increasing rate. Fig. 12 plots examples of how our approach optimizes the obstacle layout for agents. Firstly, it guides the agent crowd and sorts out the navigation randomness; for example, it guides a subset of agents to slow down and take a detour to make way for the other agents. Secondly, it minimizes the use of collision-free space and does not obstruct agent pathways.

The above results demonstrate that the obstacles are not always "bad", but instead, can play "good" guidance roles for the agents with an appropriate layout. These insights can be used for traffic design to facilitate safe and efficient navigation.

E. Real-World Experiments

We conduct real-world experiments to corroborate simulation results. We consider two scenarios, i.e., the warehouse setting in Sec. V-C and the circular setting in Sec. V-D, where the former leverages agent-environment co-optimization to improve navigation performance and the latter explores the guidance role of the environment in multi-agent de-confliction. We use the Cambridge RoboMaster robots [59] equipped with Raspberry Pi in the workspace of size $4.5\text{m}\times4.5\text{m}$, where each robot has a partially observable space with a communication range of 1m. We use ROS2 as the communication middle-ware and employs external telemetry (OptiTrack) for localization.



(a) $t = 0$ seconds **(b)** $t = 1$ seconds **(c)** $t = 4$ seconds **(d)** $t = 6$ seconds

(e) $t = 0$ seconds **(f)** $t = 2$ seconds **(g)** $t = 6$ seconds **(h)** $t = 12$ seconds

Figure 14. Real-world experiments in the warehouse setting. Robots are required to pass through obstacles towards goals, where colored lines are agent trajectories. (a-d) Performance of agent-environment coordinated optimization. (e-h) Performance of hand-designed baseline. Qualitatively, the optimized scenario in (a-d) showcases visually smoother trajectories than the hand-designed one in (e-h).

At each time, the robot captures its own state, receives neighbors' states, and generates the control output with the navigation policy in a decentralized manner. We use Model Predictive Control (MPC) as the low-level controller, which converts the control output to the wheel speed using inverse kinematics, moving the robot to its goal – see Appendix E.

Figs. 14a-14d and Figs. 15a-15d show the performance of our method in two scenarios, where the environment generative model produces compatible obstacle layouts and multi-agent policies generate smooth trajectories without collision. It outperforms the hand-designed baseline in Figs. 14e-14h and Figs. 15e-15h, which corroborates simulation results. Moreover, our method exhibits similarly good performance in real-world experiments as in numerical simulations. This is due to that the optimized environment creates more collision-free space and provides more de-confliction guidance, which ease the navigation task. It is thus more robust to the sim-to-real impact during real-world deployment. In contrast, the hand-designed baseline performs worse (e.g., lower path efficiencies, slower speed and more collisions) in real-world experiments than in numerical simulations. This is because the hand-designed environment is not optimally compatible with the navigation task, requiring more accurate execution of the navigation policy during real-world deployment, and is hence affected more seriously by the sim-to-real gap.

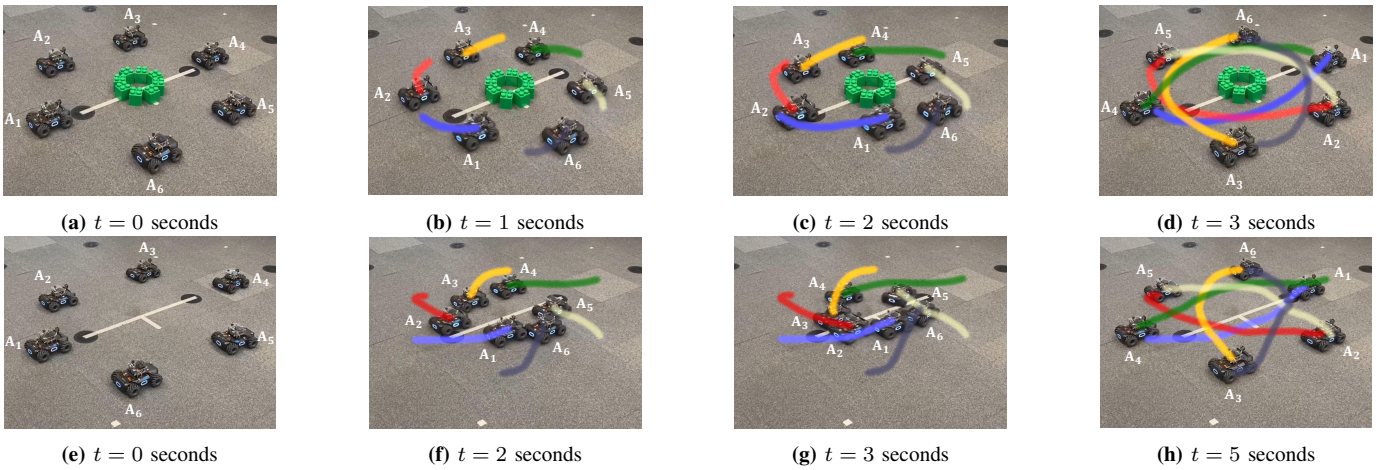
F. Discussions

In summary, the coordinated optimization method builds a symbiotic agent-environment co-existing system, where environment configurations and multi-agent policies adapt to and rely on each other to jointly improve performance. For one thing, our method optimizes the obstacle layout to create efficient obstacle-free pathways for agents and generates compatible agent trajectories for smooth navigation – see Sec. V-C. For another, our method exploits the obstacle layout (e.g., structure irregularity and non-symmetry) to prioritize created obstacle-free pathways and de-conflict agents via prioritization, which provide guidance for agents to facilitate safe navigation – see Sec. V-D. Compared to hand-designed environments, our approach achieves higher success rates and path efficiencies (SPL), moving speed (PCTSpeed), in the meantime, providing better navigation safety (NumCOLL) and acceleration comfort (DiffACC). It offers a comprehensively improved performance for multi-agent navigation and is well-applicable to various settings. Additional experiments are provided in Appendix D to further validate the proposed method.

Moreover, our framework allows high degrees of freedom for environment optimization. Specifically, in our experiments, the environment generative model allows three design choices for each obstacle: (i) The presence or absence of an obstacle; (ii) the obstacle position; (iii) the obstacle size. These reconfigurable parameters together yield a rich family of possible environment configurations, which can be used as potential candidates for different multi-agent navigation tasks.

VI. Conclusion

This paper proposed an agent-environment coordinated optimization framework for multi-agent systems. It comprises two components: *(i)* multi-agent navigation and *(ii)* environment optimization. The former seeks a navigation strategy that moves agents from origins to goals, and the latter seeks a design scheme that generates an appropriately accommodating environment (w.r.t the task). These components must be optimized jointly such that an overarching objective is achieved (in our case: improved navigation performance). We solved this problem by alternating the optimization of the multi-agent navigation policy and an environment generative model. We followed a model-free learning-based approach, and introduced a novel combination of reinforcement learning (through an actor-critic mechanism) and unsupervised learning (through policy gradient ascent) for coordinated optimization. Furthermore, we analyzed the convergence of our method by exploring its relation with an associated time-varying non-convex optimization problem and ordinary differential equations. Numerical results and real-world experiments corroborated theoretical findings and showed the benefit of our method, i.e., the agent-environment co-optimization finds agent-friendly environments and efficient navigation trajectories. Finally, we investigated the role of the environment in multi-agent navigation to show that it does not just impose physical restrictions, but can also provide guidance for agent de-confliction if designed appropriately. The latter explores an



(a) $t = 0$ seconds **(b)** $t = 1$ seconds **(c)** $t = 2$ seconds **(d)** $t = 3$ seconds

(e) $t = 0$ seconds **(f)** $t = 2$ seconds **(g)** $t = 3$ seconds **(h)** $t = 5$ seconds

Figure 15. Real-world experiments in the circular setting. Robots are required to cross-navigate towards the opposite side while avoiding collisions. (a-d) Performance of agent-environment coordinated optimization. (e-h) Performance of hand-designed default (i.e., empty) baseline. Qualitatively, the optimized scenario in (a-d) showcases visually smoother trajectories than the hand-designed one in (e-h).

inherent relationship between the environment and agents in a way previously unaddressed. Our framework has limitations when the obstacle layout of the environment is not reconfigurable or with limited flexibility for reconfiguration. It reduces to nominal multi-agent navigation when the environment is not reconfigurable and has a reduced performance improvement when the environment has limited reconfigurable flexibility.

Future work will extend the agent-environment co-optimization to life-long multi-agent navigation with multiple target positions through a problem re-formulation. We plan to model multiple targets as a sequence of waypoints, assign them as intermediate sub-goals, and move the agents to pass through all sub-goals towards the final destination. The key step is to develop scheduling strategies that assign targets to successive waypoints for each agent and re-design reward functions to learn multi-agent policies that pass through all waypoints.

Appendix A
Proof of Theorem 1

First, from Theorem 3.3 in [60], the ODE (16) has a unique solution $\boldsymbol{\theta}_o(\alpha)$ starting from the point $\boldsymbol{\theta}_o^{(0)} = \boldsymbol{\theta}_o(0)$. Then, we show the discrete parameters $\{\boldsymbol{\theta}_o^{(k)}\}_k$ of the proposed method converge to this solution $\boldsymbol{\theta}_o(\alpha)$. From Taylor's theorem, we can expand the ODE solution $\boldsymbol{\theta}_o(\alpha)$ at $\alpha^{(k+1)}$ as

$$\boldsymbol{\theta}_o(\alpha^{(k+1)}) = \boldsymbol{\theta}_o(\alpha^{(k)}) + \Delta\alpha \boldsymbol{\theta}_o'(\alpha^{(k)}) + \mathcal{O}(\Delta\alpha^2), \quad (18)$$

where $\mathcal{O}(\Delta\alpha^2)$ can be bounded by $C_h\Delta\alpha^2$ with C_h the bounding constant of the higher-order term over the finite horizon $\alpha^{(k)} \in [0, T]$. By substituting (16) into (18), we have

$$\boldsymbol{\theta}_o(\alpha^{(k+1)}) = \boldsymbol{\theta}_o(\alpha^{(k)}) - \frac{\Delta\alpha}{\eta}\nabla_{\boldsymbol{\theta}_o} g(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o(\alpha^{(k)}), \alpha^{(k)}) + \mathcal{O}(\Delta\alpha^2). \quad (19)$$

Recall the proposed method updates the generative parameters $\boldsymbol{\theta}_o^{(k)}$ by the policy gradient ascent w.r.t. the objective function $f(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o^{(k)}, \alpha^{(k)})$ of step-size $\Delta\beta = \Delta\alpha/\eta$, i.e.,

$$\begin{aligned}\boldsymbol{\theta}_o^{(k+1)} &= \boldsymbol{\theta}_o^{(k)} + \frac{\Delta\alpha}{\eta}\widetilde{\nabla}_{\boldsymbol{\theta}_o} f(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o^{(k)}, \alpha^{(k)}) \quad (20)\\ &= \boldsymbol{\theta}_o^{(k)} - \frac{\Delta\alpha}{\eta}\widetilde{\nabla}_{\boldsymbol{\theta}_o} g(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o^{(k)}, \alpha^{(k)}),\end{aligned}$$

where $\widetilde{\nabla}_{\boldsymbol{\theta}_o} f$ and $\widetilde{\nabla}_{\boldsymbol{\theta}_o} g$ represent the policy gradients that approximates the true gradients $\nabla_{\boldsymbol{\theta}_o} f$ and $\nabla_{\boldsymbol{\theta}_o} g$. By subtracting (20) from (19), we get

$$\begin{aligned}\mathbf{e}_o^{(k+1)} &= \mathbf{e}_o^{(k)} + \mathcal{O}(\Delta\alpha^2) \quad (21)\\ &+ \frac{\Delta\alpha}{\eta}\Big(\widetilde{\nabla}_{\boldsymbol{\theta}_o} g(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o^{(k)}, \alpha^{(k)}) - \nabla_{\boldsymbol{\theta}_o} g(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o(\alpha^{(k)}), \alpha^{(k)})\Big),\end{aligned}$$

where $\mathbf{e}_o^{(k)} = \boldsymbol{\theta}_o(\alpha^{(k)}) - \boldsymbol{\theta}_o^{(k)}$ is the deviation error. By using Assumption 2 and the triangle inqueality, we rewrite (21) as

$$\begin{aligned}\|\mathbf{e}_o^{(k+1)}\| &\le \|\mathbf{e}_o^{(k)}\| + C_h\eta^2\Big(\frac{\Delta\alpha}{\eta}\Big)^2 + \varepsilon\frac{\Delta\alpha}{\eta} \quad (22)\\ &+ \frac{\Delta\alpha}{\eta}\|\nabla_{\boldsymbol{\theta}_o} g(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o^{(k)}, \alpha^{(k)}) - \nabla_{\boldsymbol{\theta}_o} g(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o(\alpha^{(k)}), \alpha^{(k)})\|.\end{aligned}$$

By using Assumption 1 and $\boldsymbol{\theta}_o(\alpha^{(k)}) = \boldsymbol{\theta}_o^{(k)} + \mathbf{e}_o^{(k)}$, we have

$$\|\nabla_{\boldsymbol{\theta}_o} g(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o^{(k)}, \alpha^{(k)}) - \nabla_{\boldsymbol{\theta}_o} g(\mathbf{S}, \mathbf{D}, \boldsymbol{\theta}_o(\alpha^{(k)}), \alpha^{(k)})\| \le C_L\|\mathbf{e}_o^{(k)}\|, \quad (23)$$

and substituting (23) into (22) yields

$$\|\mathbf{e}_o^{(k+1)}\| \le (1 + \frac{\Delta\alpha}{\eta}C_L)\|\mathbf{e}_o^{(k)}\| + C_h\eta^2\Big(\frac{\Delta\alpha}{\eta}\Big)^2 + \varepsilon\frac{\Delta\alpha}{\eta}. \quad (24)$$

Then, we show by induction that

$$\|\mathbf{e}_o^{(k)}\| \le \frac{C_h\eta\Delta\alpha + \varepsilon}{C_L}\Big(\big(1 + \frac{\Delta\alpha}{\eta}C_L\big)^k - 1\Big). \quad (25)$$

For $k = 0$, we need to show $\|\mathbf{e}_o^{(0)}\| \le 0$, i.e., $\mathbf{e}_o^{(k)} = \mathbf{0}$. This is true because the initial points $\boldsymbol{\theta}_o^{(0)}$ and $\boldsymbol{\theta}_o(0)$ are the same. For iteration k, we assume (25) holds and consider iteration $k + 1$. By combining (25) with (24), we bound $\|\mathbf{e}_o^{(k+1)}\|$ by

$$\begin{aligned}&\big(1 + \frac{\Delta\alpha}{\eta}C_L\big)\frac{C_h\eta\Delta\alpha + \varepsilon}{C_L}\Big(\big(1 + \frac{\Delta\alpha}{\eta}C_L\big)^k - 1\Big) + C_h\eta^2\Big(\frac{\Delta\alpha}{\eta}\Big)^2 + \varepsilon\frac{\Delta\alpha}{\eta}\\ &= \frac{C_h\eta\Delta\alpha + \varepsilon}{C_L}\Big((1 + \frac{\Delta\alpha}{\eta}C_L)^{k+1} - 1\Big). \quad (26)\end{aligned}$$

This completes the inductive argument and proves (25).

Since the term $\Delta\alpha C_L/\eta > 0$ is positive, we have $1 + \Delta\alpha C_L/\eta \le e^{\Delta\alpha C_L/\eta}$ and thus $(1 + \Delta\alpha C_L/\eta)^k \le$

$e^{k\Delta\alpha C_L/\eta} \leq e^{\lfloor T/\Delta\alpha\rfloor \Delta\alpha C_L/\eta}$, where $k \leq \lfloor T/\Delta\alpha\rfloor$ is used. Further using the fact that $\Delta\alpha\lfloor T/\Delta\alpha\rfloor \leq T$ yields

$$\big(1+\frac{\Delta\alpha}{\eta}C_L\big)^k \leq e^{\frac{C_L}{\eta}T}. \tag{27}$$

By substituting (27) into (25), we get

$$\|\mathbf{e}_o^{(k)}\| \leq \frac{C_h\eta}{C_L}\Big(e^{\frac{C_L}{\eta}T}-1\Big)\Delta\alpha + \frac{e^{\frac{C_L}{\eta}T}-1}{C_L}\varepsilon. \tag{28}$$

The first term in (28) decreases with the step-size $\Delta\alpha$. This indicates that for any $\epsilon > 0$ and T, there exists a $\Delta\alpha$ s.t.

$$\frac{C_h\eta}{C_L}\Big(e^{\frac{C_L}{\eta}T}-1\Big)\Delta\alpha \leq \epsilon. \tag{29}$$

The second term in (28) is proportional to ε w.r.t. a constant C. By using these results in (28), we complete the proof

$$\|\mathbf{e}_o^{(K)}\| \leq \epsilon + C\varepsilon. \tag{30}$$

APPENDIX B
INFORMATION PROCESSING ARCHITECTURE

Information processing architectures parameterize the navigation policy and the generative model, allowing to represent infinite dimensional policy functions with finite parameters – see challenge (iii) of Sec. II-C. The selection of information processing architectures depends on specific problem requirements, where we consider a graph neural network (GNN) for the navigation policy due to its decentralized implementation and a deep neural network (DNN) for the environment generative model due to its strong expressive power.

A. GNN-Based Multi-Agent Policy

The multi-agent system can be modeled as a graph $\mathcal{G}$ with nodes $\{1,...,n\}$ and edges $\{(i,j)\}_{ij}$. Each node represents an agent, while each edge represents a link between two neighboring agents within communication radius. The graph structure is captured by a support matrix $\mathbf{E}$ with $[\mathbf{E}]_{ij} \neq 0$ if there is an edge between i,j or $i=j$ and $[\mathbf{E}]_{ij}=0$ otherwise. The node states are captured by a graph signal $\mathbf{X}$, which is a matrix with the ith row $[\mathbf{X}]_i$ representing the state of node i. For example, the support matrix is the adjacency matrix $\mathbf{A}$ and graph signals are positions and velocities of the agents.

GNNs are layered architectures that exploit a message passing mechanism to extract features from graph signals [61]–[63]. At each layer ℓ, a GNN consists of the message aggregation function $\mathcal{F}_{\ell,m}$ and the feature update function $\mathcal{F}_{\ell,u}$. With the input signal $\mathbf{X}_{\ell-1}$ generated at the previous layer, $\mathcal{F}_{\ell,m}$ collects the signal values of the neighboring nodes and generates the intermediate features as

$$[\mathbf{U}_\ell]_i = \sum_{j\in\mathcal{N}_i} \mathcal{F}_{\ell,m}\big([\mathbf{X}_{\ell-1}]_i, [\mathbf{X}_{\ell-1}]_j, [\mathbf{E}]_{ij}\big),\ i=1,...,n, \tag{31}$$

where $\mathcal{N}_i$ is the neighbor set. These intermediate features are processed by $\mathcal{F}_{\ell,u}$ to generate the output signal as

$$[\mathbf{X}_\ell]_i = \mathcal{F}_{\ell,u}\Big([\mathbf{X}_{\ell-1}]_i, [\mathbf{U}_\ell]_i\Big),\ i=1,...,n. \tag{32}$$

The GNN inputs are the node states $\mathbf{X}$ and the GNN output is the last layer output $\mathbf{X}_L$. We define the GNN as a nonlinear mapping $\mathbf{\Phi}(\mathbf{X},\mathbf{E},\boldsymbol{\theta}_a)$ where the architecture parameters $\boldsymbol{\theta}_a$ collect all function parameters of $\{\mathcal{F}_{\ell,m},\mathcal{F}_{\ell,u}\}_{\ell=1}^L$.

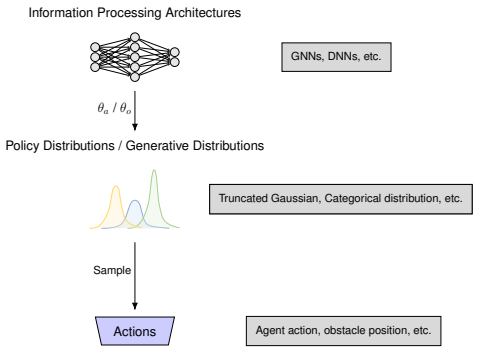


Figure 16. General framework of parameterization. The information processing architecture parameterizes the navigation policy or the generative model. These parameters determine the policy distributions of the agents or the generative distribution of the environment, and the distribution samples the agent actions or the obstacle layout.

We parameterize the navigation policy with GNNs because they allow for a *decentralized implementation*. Specifically, the message aggregation in (31) requires only signal values of the neighboring nodes and the latter can be obtained by communications, while the feature update in (32) is a local operation and does not affect the decentralized nature (e.g., see [46], [64], [65]); hence, the output signal can be computed at each node locally with neighborhood information only.

B. DNN-Based Generative Model

The generative model generates the obstacle layout based on the navigation task, i.e., the initial and goal positions, before agents start to move. In this context, we can deploy *centralized* information processing architectures to parameterize the generative model. DNNs are well-suited candidates that have achieved success in a wide array of applications [66]–[68]. Specifically, DNNs comprise multiple layers, where each layer consists of a linear operation and a nonlinearity. At layer ℓ, the input signal $\mathbf{X}_{\ell-1} \in \mathbb{R}^{F_{\ell-1}\times F}$ is processed by a linear operator $\mathbf{\Gamma}_\ell \in \mathbb{R}^{F_\ell \times F_{\ell-1}}$ and passed through a nonlinearity $\sigma(\cdot)$ to generate $\mathbf{X}_\ell \in \mathbb{R}^{F_\ell \times F}$, where F_ℓ is the number of units at layer ℓ and F is the number of features at each unit, i.e.,

$$\mathbf{X}_0 = \mathbf{X},\ \mathbf{X}_\ell = \sigma\big(\mathbf{\Gamma}_\ell \mathbf{X}_{\ell-1}\big) \text{ for } \ell = 1,...,L, \tag{33}$$

where the initial and goal positions $\mathbf{X}=[\mathbf{S},\mathbf{D}]$ are the DNN inputs, the last layer output $\mathbf{X}_L = \mathbf{\Phi}_o(\mathbf{X},\boldsymbol{\theta}_o)$ is the DNN output, and the architecture parameters $\boldsymbol{\theta}_o$ are the weights of all $\{\mathbf{\Gamma}_\ell\}_{\ell=1}^L$. We remark that DNNs offer a near-universal parameterization, i.e., they can approximate any function to any desired degree of accuracy given sufficiently large layers and features, providing a strong representational capacity [69].

C. Policy Distribution

We consider the GNN (or DNN) output $\mathbf{X}_L$ as the parameters of the policy (or generative) distribution and sample the agent actions $\mathbf{U}_a$ (or obstacle layout $\mathcal{O}$) from the distribution instead of directly computing them. This not only allows to train the parameters $(\boldsymbol{\theta}_a,\boldsymbol{\theta}_o)$ with policy gradient in a model-free manner [cf. (8)-(9)], but also makes the generated actions

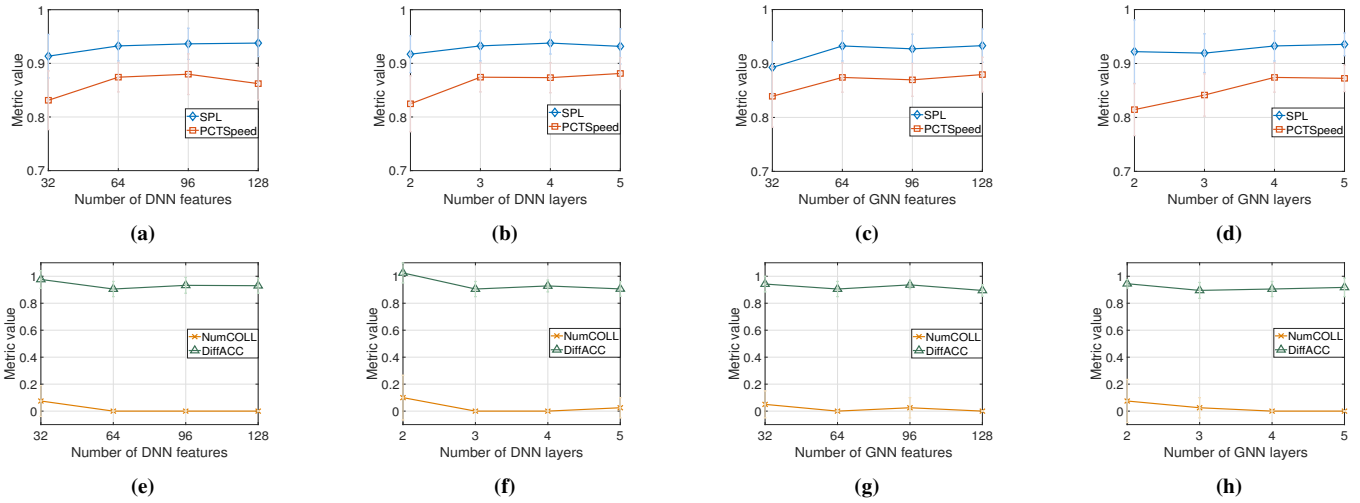


Figure 17. Performance of agent-environment coordinated optimization with different layers and features of DNNs for multi-agent navigation policies and GNNs for environment generative models. (a-d) SPL and PCTSpeed with different features and layers of DNNs and GNNs, where a higher value represents a better performance, i.e., higher success rate, path efficiency and speed. (e-h) NumCOLL and DiffACC with different features and layers of DNNs and GNNs, where a lower value represents a better performance, i.e., more safety and comfort.

satisfy the allowable space, i.e., $\mathbf{U}_a \in \mathcal{U}_a$ and $\mathcal{O} \in \mathcal{P}_o$ in (1). Specifically, these constraints restrict the space of feasible solutions, which are usually difficult to satisfy and require specific post-processing techniques. We overcome this issue by selecting different policy / generative distributions to satisfy different constraints – see details in the following examples.

Truncated Gaussian distribution. If the action is within a bounded interval $[0, C]$, we select the distribution as a truncated Gaussian distribution. Specifically, it is a Gaussian distribution with mean μ and variance σ^2, and lies in the interval $[0, C]$ with the probability density function

$$f(x, \mu, \sigma | 0, C) = \frac{1}{\sigma} \frac{\Psi\left(\frac{x-\mu}{\sigma}\right)}{\mathbf{\Psi}\left(\frac{C-\mu}{\sigma}\right) - \mathbf{\Psi}\left(\frac{-\mu}{\sigma}\right)}, \tag{34}$$

where $\Psi(x)$ is the probability density function and $\mathbf{\Psi}(x)$ is the cumulative distribution function of the standard Gaussian distribution. The distribution parameters μ and σ are determined by the GNN (or DNN) output $\mathbf{X}_L$.

Categorical distribution. If the action is to select one value from a set of potential values, e.g., the agent takes one of K discrete actions and the obstacle either exists or not, we select the distribution as a categorical distribution. Specifically, it takes one of K categories and the probability of each one is

$$f(x = k) = p_k \text{ s.t. } \sum_{k=1}^{K} p_k = 1. \tag{35}$$

The distribution parameters $\{p_k\}_{k=1}^K$ are determined by the GNN (or DNN) output $\mathbf{X}_L$. For comprehensive restrictions including both (34) and (35), we select a combination of two distributions. Fig. 16 illustrates the aforementioned framework.

The agent-environment co-optimization can be applied to more agents and obstacles because GNNs allow for a decentralized implementation that is scalable to larger multi-agent systems and DNNs have a strong expressive power to handle larger environment settings. As the first stride in this research direction, our work aims to formalize the problem setting and develop a general solution framework, to explore the relationship between agents and environment and to lay a foundation to promote extensions. For future exploration, we can consider simulators that allow to leverage GPUs (e.g., NVIDIA Isaac Sim) to simulate a growing large number of robots and obstacles with physical properties (e.g., inertia, friction, wheel torque, etc.), and evaluate the agent-environment co-optimization for larger-scale problems.

Appendix C
Performance Metrics

This paper considers four metrics, i.e., SPL, PCTSpeed, NumCOLL and DiffACC. These metrics include the information about agents' success, path efficiencies, speed, safety and comfort, which provide a comprehensive performance evaluation to show the effectiveness of the proposed method.

SPL is the gold standard to measure the performance of robot navigation in the literature [54]–[56] defined as

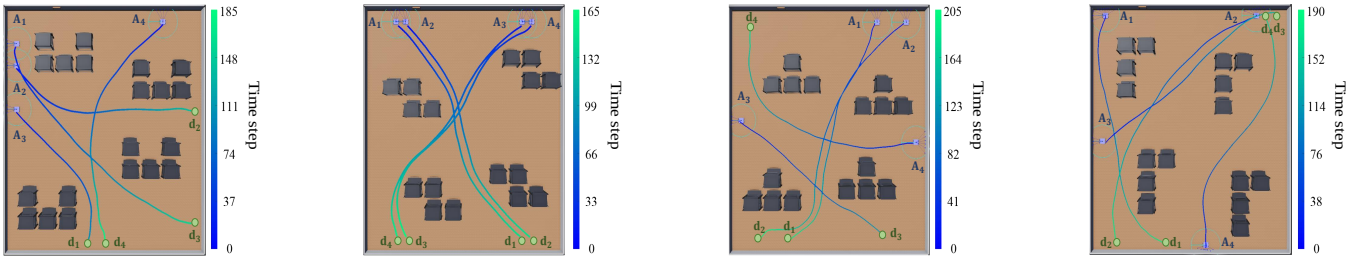
$$\text{SPL} = \frac{1}{n} \sum_{i=1}^{n} s_i \frac{\ell_i}{\max\{p_i, \ell_i\}}, \tag{36}$$

where s_i is a binary indicator for the success of agent A_i, p_i is the traveled distance and ℓ_i is the shortest distance. SPL leverages $\{s_i\}_{i=1}^n$ to represent whether navigation tasks of $\{A_i\}_{i=1}^n$ are successful, and the distance ratios $\left\{\ell_i / \max\{p_i, \ell_i\}\right\}_{i=1}^n$ to represent the path efficiencies of successful agents. It is ready to see that: (1) SPL is proportional to the success rate, i.e., the higher the success rate, the larger the SPL; (2) Given the same success rate, SPL is proportional to the path efficiency, i.e., the higher the path efficiency, the larger the SPL. Therefore, it is a stringent measure that can be used as a primary quantifier in comparing the navigation performance.

PCTSpeed, **NumCOLL** and **DiffACC** are defined as

$$\text{PCTSpeed} = \frac{1}{nT} \sum_{i=1}^{n} \sum_{t=1}^{T} \frac{\|\mathbf{v}_i^{(t)}\|}{\|\mathbf{v}_{\max}\|}, \tag{37}$$

$$\text{NumCOLL} = \sum_{i=1}^{n} \frac{N_i}{n}, \ \text{DiffACC} = \sum_{i=1}^{n} \sum_{t=1}^{T} \frac{\|\mathbf{a}_i^{(t)} - \mathbf{a}_i^{(t-1)}\|}{nT}, \tag{38}$$



(a) **(b)** **(c)** **(d)**

Figure 18. Examples of agent-environment co-optimization with non-convex irregular obstacles of different shapes. Blue robots $\{A_i\}_{i=1}^4$ represent agents at initial positions and green circles $\{\mathbf{d}_i\}_{i=1}^4$ represent goal positions. Colored lines from blue to green are agent trajectories and the color bar represents the time scale. (a) U-shaped obstacles. (b) Z-shaped obstacles. (c) $\perp$-shaped obstacles. (d) Γ-shaped obstacles.

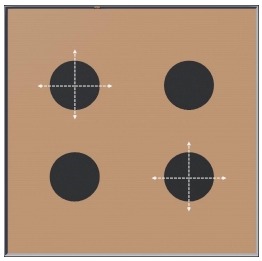


Figure 19. Environment for evaluating the effect of obstacle / agent size on the agent-environment co-optimization. The obstacle positions are re-configurable along x- and y-axes in a neighborhood of 5×5.

where $\mathbf{v}_i^{(t)}$ is the velocity of agent A_i at time step t, $\mathbf{v}_{\max}$ is the maximal velocity, N_i is the number of collisions, $\mathbf{a}_i^{(t)}$ is the acceleration of A_i, and $\mathbf{a}_i^{(0)} = \mathbf{0}$ by default. PCTSpeed leverages the speed ratios to represent how fast agents move along trajectories. NumCOLL characterizes how safe agents are w.r.t. each other and obstacles along trajectories, while DiffACC computes the expected change of agents' accelerations, which measures how comfortable agents are along trajectories.

SPL and PCTSpeed normalize the metric value to $[0, 1]$ with a unified exposition for a joint comparison, where a higher value represents a better performance. NumCOLL and DiffACC are auxiliary metrics, where a lower value represents a better performance (i.e., more safety and comfort).

APPENDIX D
ADDITIONAL EXPERIMENTS

We conduct additional experiments to further validate our method. First, we evaluate it with different hyperparameters, i.e., features and layers of DNNs and GNNs for the environment generative model and multi-agent navigation policies. Second, we consider irregular obstacles with U, Z, $\perp$ and Γ shapes, to show the general applicability of our method. Third, we characterize the effect of obstacle / agent size on the performance of our method. Lastly, we investigate the role of inter-agent collision penalty in guiding agent de-confliction, in addition to the obstacle guidance stated in Sec. V-D.

A. Hyperparameter Sensitivity

In our experiments of Sec. V, we consider a 3-layered DNN and a 1-layered GNN, where message aggregation and feature update functions are 4-layered MLPs with 64 units and ReLU nonlinearity per layer. This is a common selection, which has already exhibited superior performance. To further study hyperparameter effects, we evaluate our approach with different layers and features of DNNs and GNNs in Fig. 17[3].

We see that SPL / PCTSpeed increases and NumCOLL / DiffACC decreases with the number of layers or features for both DNNs and GNNs. This is because more layers or features yield a stronger representational capacity for parameterization and thus, an improved performance for coordinated optimization. When the number of features or layers is large, the representational capacity reaches a saturated level and the performance fluctuates around the optimal value. We remark that the proposed method achieves a good performance even with a small number of layers (e.g., 2 layers) or features (e.g., 32 features). This indicates that the proposed method is not sensitive to the selection of neural network hyperparameters.

B. Obstacle Irregularity

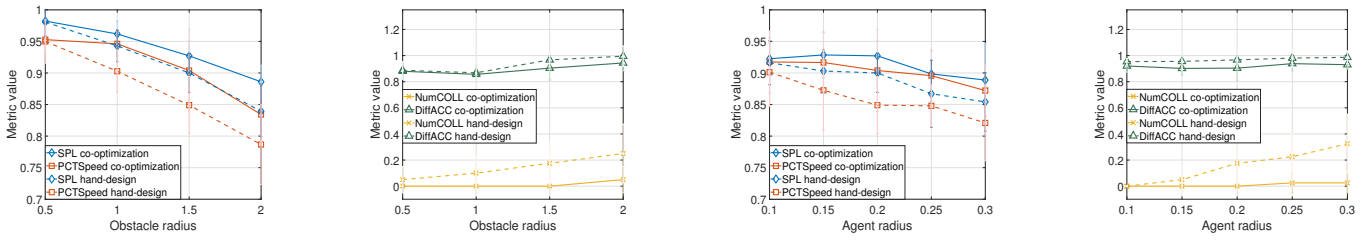
We consider rectangular and circular obstacles in Sec. V, following the standard setting of multi-agent navigation in the literature [70]–[72]. Our method can be extended to irregular obstacles of other shapes. Given any design choices of irregular obstacles, we can consider these re-configurable parameters as outputs of the environment generative model and carry out the coordinated optimization in the same manner.

To validate this extension, we evaluate the proposed method for non-convex obstacles with different irregular shapes, which include U-shaped, Z-shaped, $\perp$-shaped and Γ-shaped obstacles. Fig. 18 shows examples of the agent-environment co-optimization in these scenarios. We see that the proposed method achieves good performance for non-convex irregular obstacles as well as regular ones. The environment generative model produces compatible obstacle layouts and the multi-agent navigation policy generates smooth agent trajectories, yielding superior performance in a joint manner.

C. Agent and Obstacle Size

From Sec. V, we see that more obstacles result in more complex navigation scenarios, require more compatible environments for multi-agent navigation, and emphasize the performance improvement introduced by agent-environment co-optimization (from Fig. 6a to Fig. 6c). Moreover, more agents

[3]We maintain a 1-layered GNN for an efficient decentralized implementation with only 1-hop communication, and evaluate different layers and features of message aggregation and feature update functions.



(a) **(b)** **(c)** **(d)**

Figure 20. Performance of agent-environment coordinated optimization (solid line) and hand-designed baseline (dashed line) with different obstacle and agent sizes. (a) SPL and PCTSpeed with different obstacle sizes, where a higher value represents a better performance. (b) NumCOLL and DiffACC with different obstacle sizes, where a lower value represents a better performance. (c) SPL and PCTSpeed with different agent sizes. (d) NumCOLL and DiffACC with different agent sizes.

lead to more cluttered navigation scenarios, require more guidance from obstacles for de-confliction, and increase the performance improvement obtained by agent-environment co-optimization (from Fig. 8a to Fig. 8c). This subsection studies the influence of obstacle / agent size on the co-optimization performance. Specifically, we consider the environment of size 12×12, where obstacle positions can be re-configured along x- and y-axes in a neighborhood of size 5×5 – see Fig. 19.

Fig. 20 shows the results with obstacle radius from 0.5 to 2 and agent radius from 0.1 to 0.3. Our method outperforms the hand-designed baseline consistently in all cases, and the performance improvement introduced by agent-environment co-optimization increases with the obstacle or agent size. This is because a larger obstacle or agent size leads to a more challenging navigation scenario and requires a more compatible environment for agents, highlighting the importance of establishing a symbiotic agent-environment co-existing system.

D. Inter-Agent Collision Penalty

We show in Sec. V-D that obstacles can have positive effects that guide agents to de-conflict for safe navigation. On the other hand, each agent can be viewed as a dynamic "obstacle" to the other agents and may play a similar guidance role. To investigate this factor, we consider the circular scenario with no obstacle in Sec. V-D, and train multi-agent navigation policies with different collision penalties between agents in the range of $[1, 12.5]$. Fig. 21 shows the results.

NumCOLL decreases with the increasing of inter-agent collision penalty. This is expected as a larger penalty leads to more conservative behaviors and the latter facilitates agent de-confliction for safe navigation. However, this results in worse navigation performance, i.e., lower SPL and PCTSpeed, because the agents focus more on safety than goal reaching. Moreover, the co-optimization method consistently outperforms the case of the largest penalty, with a lower NumCOLL and larger SPL, PCTSpeed. These highlight the importance of obstacle guidance for safe multi-agent navigation.

We conclude that while inter-agent collision is beneficial for de-confliction, it may not be sufficient to guide agents and involving obstacles is helpful, especially in cluttered scenarios. Moreover, there exist scenarios where obstacles have already existed and are not allowed to be removed. For example, in the warehouse setting, there must exist shelves to store packages and we cannot remove them from the space. In these cases, the proposed method can optimize the layout of the existing obstacles to guide agents and improve performance.

APPENDIX E
CONTROL DYNAMICS

In this section, we provide details about control dynamics considered in our experiments. Specifically, when training multi-agent navigation policies, we assume single integrator systems with constrained velocity inputs, i.e.,

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{v}(t), \text{ s.t. } \mathbf{v}(t) \in \mathcal{V}, \tag{39}$$

where $\mathbf{x}(t)$ is the 2-D state vector, $\mathbf{A}$ a 2 by 2 matrix with all zeroes, $\mathbf{B}$ a 2 by 2 identity matrix, $\mathbf{v}(t)$ the velocity vector, $\mathcal{V}$ the action space. This is a standard setting in the study of multi-agent navigation, and has been widely used in [73]–[75].

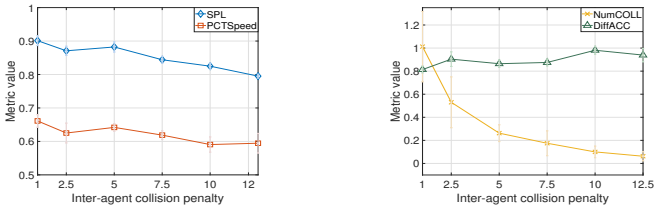
When deploying multi-agent policies in real-world experiments (or Webots simulations), we have omnidirectional robots with nonlinear dynamics, and use Model Predictive Control (MPC) and inverse kinematics as the low-level controller to convert the output of navigation policies to the speed of wheels. Specifically, consider the discrete-time linear time-invariant (LTI) system $\mathbf{x}[k+1] = \tilde{\mathbf{A}}\mathbf{x}[k] + \tilde{\mathbf{B}}\mathbf{u}[k]$, where $\mathbf{x}[k]$ is the state vector at time step k, $\mathbf{u}[k]$ is the control input, $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are state transition matrix and control input matrix derived from (39). The MPC minimizes a cost function over a finite prediction horizon M, subject to system dynamics and control input constraints. In particular, the cost function is [76], [77]

$$\begin{aligned} J = \sum_{i=0}^{M-1} \big(\mathbf{x}[k+i|k]^T \mathbf{Q}\mathbf{x}[k+i|k] + \mathbf{u}[k+i|k]^T \mathbf{R}\mathbf{u}[k+i|k]\big) \\ + \mathbf{x}[k+M|k]^T \mathbf{P}\mathbf{x}[k+M|k], \end{aligned} \tag{40}$$

where $\mathbf{x}[k+i|k]$ ($\mathbf{u}[k+i|k]$) is the predicted state (control input) at time step $k+i$ given the state (control input) at time step k, $\mathbf{Q}$ is the state weighting matrix, $\mathbf{R}$ is the control weighting matrix, and $\mathbf{P}$ is the terminal state weighting matrix. The MPC problem can then be formulated as

$$\begin{aligned} \min_{\{\mathbf{u}[k],\ldots,\mathbf{u}[k+M-1]\}} \quad & J \\ \text{s.t.} \quad & \mathbf{x}[k+i+1|k] = \tilde{\mathbf{A}}\mathbf{x}[k+i|k] + \tilde{\mathbf{B}}\mathbf{u}[k+i|k], \\ & \mathbf{u}[k+i|k] \in \mathcal{U}, \quad i = 0, \ldots, M-1, \end{aligned} \tag{41}$$

where $\mathcal{U}$ is the action space of control input. We consider off-the-shelf solver CVXPY [78] that leverages Primal-Dual Interior-Point optimization method to solve problem (41).



(a) **(b)**

Figure 21. Navigation performance in the circular setting of 16 agents, with no obstacle and different inter-agent collision penalties. (a) SPL and PCTSpeed. (b) NumCOLL and DiffACC.

Next, we convert the obtained control input $\mathbf{u} = [\mathbf{u_x}, \mathbf{u_y}]^\top$ to the wheel speed. Since omnidirectional robots have squared base, let l be the length from its center to the edge. The velocity command of wheels can be computed as [79], [80]

$$\begin{bmatrix} \mathbf{v}_{w_1} \\ \mathbf{v}_{w_2} \\ \mathbf{v}_{w_3} \\ \mathbf{v}_{w_4} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & l \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & l \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & l \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & l \end{bmatrix} \begin{bmatrix} \mathbf{u}_x \\ \mathbf{u}_y \\ \omega \end{bmatrix}, \tag{42}$$

where $\omega = 0$ is the angular velocity in our case. This indicates that our method can handle actual nonlinear dynamics by abstracting the nonlinear dynamical system as a single integrator system to train navigation policies and leveraging the low-level controller to convert policy outputs to robot actions for real-world deployment. Such an abstraction technique has been widely used in the literature of omnidirectional robots and also extended to drone systems [81]–[83].

Lastly, we remark that the performance of the proposed method can be further improved by establishing high-fidelity physics models for real-world systems and integrating the latter into the coordinated optimization procedure. Specifically, we can characterize the actual dynamical system through system identification methods [84]–[86]. In the phase of multi-agent navigation, we can leverage the high-fidelity model to determine the state transition function to train the navigation policy with reinforcement learning. In the phase of environment optimization, we can evaluate the multi-agent policy in the generated environment with the high-fidelity model and update the generative model with the corresponding reward. It accounts for realistic dynamics when updating multi-agent policies and environment generative models, and has potential of further improving the co-optimization performance.

REFERENCES

[1] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE International Conference on Robotics and Automation*, 2008.

[2] V. Desaraju and J. How, "Decentralized path planning for multi-agent teams with complex constraints," *Autonomous Robots*, vol. 32, pp. 385–403, 2012.

[3] T. Standley and R. Korf, "Complete algorithms for cooperative pathfinding problems," in *International Joint Conference on Artificial Intelligence*, 2011.

[4] W. Wu, S. Bhattacharya, and A. Prorok, "Multi-robot path deconfliction through prioritization by path prospects," in *IEEE International Conference on Robotics and Automation*, 2020.

[5] M. Everett, Y. Chen, and J. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[6] Z. Gao, G. Yang, and A. Prorok, "Online control barrier functions for decentralized multi-agent navigation," in *IEEE International Symposium on Multi-Robot and Multi-Agent Systems*, 2023.

[7] N. Mani, V. Garousi, and B. Far, "Search-based testing of multi-agent manufacturing systems for deadlocks based on models," *International Journal on Artificial Intelligence Tools*, vol. 19, pp. 417–437, 2010.

[8] A. Ruderman et al., "Uncovering Surprising Behaviors in Reinforcement Learning via Worst-case Analysis," in *International Conference on Learning Representations Workshop*, 2019.

[9] Q. Wang, R. McIntosh, and M. Brain, "A new-generation automated warehousing capability," *International Journal of Computer Integrated Manufacturing*, vol. 23, no. 6, pp. 565–573, 2010.

[10] H. Bier, "Robotic building (s)," *Next Generation Building*, vol. 1, 2014.

[11] M. Bellusci, N. Basilico, and F. Amigoni, "Multi-agent path finding in configurable environments," in *International Conference on Autonomous Agents and MultiAgent Systems*, 2020.

[12] L. Custodio and R. Machado, "Flexible automated warehouse: a literature review and an innovative framework," *International Journal of Advanced Manufacturing Technology*, vol. 106, pp. 533–558, 2020.

[13] M. Čáp et al., "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 835–849, 2015.

[14] H. Zhang, Y. Chen, and D. Parkes, "A general approach to environment design with one agent," in *International Joint Conference on Artificial Intelligence*, 2009.

[15] S. Keren, A. Gal, and E. Karpas, "Goal recognition design for non-optimal agents," in *AAAI Conference on Artificial Intelligence*, 2015.

[16] M. Čáp, J. Vokřínek, and A. Kleiner, "Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures," in *International Conference on Automated Planning and Scheduling*, 2015.

[17] A. Kulkarni et al., "Designing environments conducive to interpretable robot behavior," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[18] Z. Gao and A. Prorok, "Environment optimization for multi-agent navigation," in *International Conference on Robotics and Automation*, 2022.

[19] Z. Gao and A. Prorok, "Constrained environment optimization for prioritized multi-agent navigation," *IEEE Open Journal of Control Systems*, 2023.

[20] P. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

[21] S. Karma et al., "Use of unmanned vehicles in search and rescue operations in forest fires: Advantages and limitations observed in a field trial," *International Journal of Disaster Risk Reduction*, vol. 13, pp. 307–312, 2015.

[22] Z. Drezner and G. Wesolowsky, "Selecting an optimum configuration of one-way and two-way routes," *Transportation Science*, vol. 31, no. 4, pp. 386–394, 1997.

[23] D. Johnson and J. Wiles, "Computer games with intelligence," in *IEEE International Conference on Fuzzy Systems*, 2001.

[24] A. Clark, *Being there: Putting brain, body, and world together again*, MIT press, 1998.

[25] T. Tanaka and H. Sandberg, "Sdp-based joint sensor and controller design for information-regularized optimal lqg control," in *Conference on Decision and Control*, 2015.

[26] S. Tatikonda and S. Mitter, "Control under communication constraints," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1056–1068, 2004.

[27] V. Tzoumas et al., "Sensing-constrained lqg control," in *American Control Conference*, 2018.

[28] H. Lipson and J. Pollack, "Automatic design and manufacture of robotic lifeforms," *Nature*, vol. 406, no. 6799, pp. 974–978, 2000.

[29] N. Cheney et al., "Scalable co-optimization of morphology and control in embodied machines," *Journal of the Royal Society Interface*, vol. 15, no. 143, pp. 20170937, 2018.

[30] C. Schaff et al., "Jointly learning to construct and control agents using deep reinforcement learning," in *International Conference on Robotics and Automation*, 2019.

[31] T. Gabel and M. Riedmiller, "Joint equilibrium policy search for multi-agent scheduling problems," in *German Conference on Multiagent System Technologies*, 2008.

[32] C. Zhang and J. Shah, "Co-optimizating multi-agent placement with task assignment and scheduling.," in *International Joint Conference on Artificial Intelligence*, 2016.

[33] R. Jain, P. Panda, and S. Subramoney, "Cooperative multi-agent reinforcement learning-based co-optimization of cores, caches, and on-chip network," *ACM Transactions on Architecture and Code Optimization*, vol. 14, no. 4, pp. 1–25, 2017.

[34] U. Ali et al., "Motion-communication co-optimization with cooperative load transfer in mobile robotics: An optimal control perspective," *IEEE Transactions on Control of Network Systems*, vol. 6, pp. 621–632, 2018.

[35] H. Jaleel and J. Shamma, "Decentralized energy aware co-optimization of mobility and communication in multiagent systems," in *IEEE Conference on Decision and Control*, 2016.

[36] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robotics and Autonomous Systems*, vol. 41, no. 2-3, pp. 89–99, 2002.

[37] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[38] I. Gur et al., "Adversarial environment generation for learning to navigate the web," *arXiv preprint arXiv:2103.01991*, 2021.

[39] K. K. Hauser, "Minimum constraint displacement motion planning.," in *Robotics: Science and Systems*. Berlin, Germany, 2013, vol. 6, p. 2.

[40] K. Hauser, "The minimum constraint removal problem with three robotics applications," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 5–17, 2014.

[41] M. Liu et al., "Task and path planning for multi-agent pickup and delivery," in *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2019.

[42] T. Yamauchi, Y. Miyashita, and T. Sugawara, "Path and action planning in non-uniform environments for multi-agent pickup and delivery tasks," in *European Conference on Multi-Agent Systems*, 2021.

[43] T. Yamauchi, Y. Miyashita, and T. Sugawara, "Standby-based deadlock avoidance method for multi-agent pickup and delivery tasks," in *International Conference on Autonomous Agents and Multiagent Systems*, 2022.

[44] A. Jiménez, V. García-Díaz, and S. Bolaños, "A decentralized framework for multi-agent robotic systems," *Sensors*, vol. 18, no. 2, pp. 417, 2018.

[45] K. Sivanathan et al., "Decentralized motion planning for multi-robot navigation using deep reinforcement learning," in *IEEE International Conference on Intelligent Sustainable Systems*, 2020.

[46] E. Tolstaya et al., "Learning decentralized controllers for robot swarms with graph neural networks," in *Conference on Robot Learning*, 2020.

[47] X. Ji et al., "Decentralized, unlabeled multi-agent navigation in obstacle-rich environments using graph neural networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021.

[48] R. Sutton et al., "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, 1999.

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

[50] I. Grondman et al., "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 42, no. 6, pp. 1291–1307, 2012.

[51] Y. Ding, J. Lavaei, and M. Arcak, "Escaping spurious local minimum trajectories in online time-varying nonconvex optimization," in *IEEE American Control Conference*, 2021.

[52] J. Schulman et al., "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[53] Cyberbotics, "Webots documentation: Contact properties," 2024.

[54] P. Anderson et al., "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[55] T. Gervet et al., "Navigating to objects in the real world," *Science Robotics*, vol. 8, no. 79, pp. eadf6991, 2023.

[56] X. Wang et al., "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2019.

[57] H. Bellem et al., "Objective metrics of comfort: Developing a driving style for highly automated vehicles," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 41, pp. 45–54, 2016.

[58] L. Eboli, G. Mazzulla, and G. Pungillo, "Measuring bus comfort levels by using acceleration instantaneous values," *Transportation Research Procedia*, vol. 18, pp. 27–34, 2016.

[59] J. Blumenkamp et al., "The cambridge robomaster: An agile multi-robot research platform," in *International Symposium on Distributed Autonomous Robotic Systems*, 2024.

[60] H. K. Khalil, *Nonlinear systems*, Upper Saddle River, 2002.

[61] F. Scarselli et al., "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[62] P. Veličković et al., "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[63] Z. Gao, E. Isufi, and A. Ribeiro, "Stochastic graph neural networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 4428–4443, 2021.

[64] Q. Li et al., "Graph neural networks for decentralized multi-robot path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[65] Z. Gao, F. Gama, and A. Ribeiro, "Wide and deep graph neural network with distributed online learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 3862–3877, 2022.

[66] W. Liu et al., "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[67] C. Sánchez-Sánchez and D. Izzo, "Real-time optimal control via deep neural networks: study on landing problems," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 5, pp. 1122–1135, 2018.

[68] Z. Gao, M. Eisen, and A. Ribeiro, "Optimal wdm power allocation via deep learning for radio on free space optics systems," in *IEEE Global Communications Conference*, 2019.

[69] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[70] Z. Ismail, N. Sariff, and E. Hurtado, "A survey and analysis of cooperative multi-agent robot systems: challenges and directions," *Applications of Mobile Robots*, vol. 5, pp. 8–14, 2018.

[71] D. Hildreth and S. Guy, "Coordinating multi-agent navigation by learning communication," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 2, no. 2, pp. 1–17, 2019.

[72] C. Yu, H. Yu, and S. Gao, "Learning control admissibility models with graph neural networks for multi-agent navigation," in *Conference on Robot Learning*, 2023.

[73] L. Martinović, Ž. Zečević, and B. Krstajić, "Cooperative tracking control of single-integrator multi-agent systems with multiple leaders," *European Journal of Control*, vol. 63, pp. 232–239, 2022.

[74] A. Amirkhani and A. H. Barshooi, "Consensus in multi-agent systems: a review," *Artificial Intelligence Review*, vol. 55, no. 5, pp. 3897–3935, 2022.

[75] F. Sun et al., "Group consensus for fractional-order heterogeneous multi-agent systems under cooperation-competition networks with time delays," *Communications in Nonlinear Science and Numerical Simulation*, vol. 133, pp. 107951, 2024.

[76] B. Zhu, K. Guo, and L. Xie, "A new distributed model predictive control for unconstrained double-integrator multiagent systems," *IEEE Transactions on Automatic Control*, vol. 63, pp. 4367–4374, 2018.

[77] T. Ghandriz et al., "Trajectory-following and off-tracking minimisation of long combination vehicles: A comparison between nonlinear and linear model predictive control," *Vehicle System Dynamics*, vol. 62, no. 2, pp. 277–310, 2024.

[78] S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[79] I. Siradjuddin et al., "A general inverse kinematic formulation and control schemes for omnidirectional robots," *Engineering Letters*, vol. 29, pp. 1, 2022.

[80] W. Li et al., "Motion planning for omnidirectional wheeled mobile robot by potential field method," *Journal of Advanced Transportation*, vol. 2017, no. 1, pp. 4961383, 2017.

[81] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *IEEE Transactions on Robotics*, vol. 20, no. 5, pp. 865–875, 2004.

[82] K. Leahy et al., "Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints," *Autonomous Robots*, vol. 40, pp. 1363–1378, 2016.

[83] A. Shankar, S. Elbaum, and C. Detweiler, "Freyja: A full multirotor system for agile & precise outdoor flights," in *IEEE International Conference on Robotics and Automation*, 2021.

[84] T. Martin, N. Umetani, and B. Bickel, "Omniad: data-driven omni-directional aerodynamics," *ACM Transactions on Graphics*, vol. 34, no. 4, pp. 1–12, 2015.

[85] N. Seegmiller and A. Kelly, "High-fidelity yet fast dynamic models of wheeled mobile robots," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 614–625, 2016.

[86] Q. Khan and R. Akmeliawati, "Review on system identification and mathematical modeling of flapping wing micro-aerial vehicles," *Applied Sciences*, vol. 11, no. 4, pp. 1546, 2021.