SRL-MPC: Shape-Aware Reinforcement Learned Model Predictive Control

Ruihua Han[1] **Rui Gao**[2] **Zhe Liu**[1] **Xinyi Wang**[3] **Chang Chen**[1]
Shuai Wang[4] **Qi Hao**[2] **Jia Pan**[1] **Hengshuang Zhao**[†1]

[1]The University of Hong Kong
[2]Research Institute of Trustworthy Autonomous Systems,
Southern University of Science and Technology
[3]University of Michigan [4]Shenzhen Institutes of Advanced Technology

Project: `https://hku-sail.github.io/srl_mpc`

Abstract

Safe and efficient shape-aware navigation in heterogeneous crowds and robot fleets remains challenging. Traditional approaches often assume homogeneous robots, sparse workspaces, simplified geometry, offline computation, or handcrafted parameters to make the problem tractable, which limits their deployment in dense crowd scenarios. Toward this end, we propose Shape-Aware Reinforcement Learned Model Predictive Control (**SRL-MPC**), a method for efficient and adaptive navigation in crowds with heterogeneous shapes using explicit footprint models. To encourage shape-aware collision avoidance, we formulate high-order control barrier function (HOCBF) constraints from geometric separation features (GSFs) based on support function transformation. A reinforcement learning (RL) framework then learns a neural policy that reads GSFs and outputs real-time MPC parameter updates, enabling the MPC solver to adapt to neighboring crowd geometries. The key advantage of SRL-MPC is that it retains the explicit geometric and kinematic structure and generalizability of MPC while integrating the adaptability and intelligence of RL. Experiments in randomized crowd scenarios, out-of-distribution settings, and real-robot trials evaluate the performance of SRL-MPC. The results show dramatically improved success rates in dense scenarios through our combination of learned parameter adaptation and shape-aware HOCBF residual penalties.

1 Introduction

Safe navigation in crowds with heterogeneous shapes is a central capability for robots operating in shared spaces with other robots and humans [9]. Here, shape-aware navigation refers to explicitly incorporating occupied body geometry into motion planning and control. The problem becomes difficult when a large number of robots must satisfy safety, efficiency, and kinematic constraints simultaneously. Existing approaches, including velocity obstacle (VO)-based methods [32, 11, 25], optimization-based methods [41, 24, 21, 22], and reinforcement learning (RL)-based methods [4, 13, 33, 38], have shown promising results, but they often rely on assumptions such as homogeneous robots, sparse workspaces, or simplified circular shape representations [32, 4, 14, 31], as well as offline computation or repeatedly tuned parameters for specific scenarios [41, 22, 20]. These assumptions make the problem tractable, but they also limit deployment in dense and dynamic crowds, where robots may get stuck or collide if the geometric information is not considered explicitly.

The core difficulty is to avoid collisions with dense, dynamic, and arbitrarily shaped objects in real time. RL-based methods can learn strong neural policies for specific scenarios, but they are prone to

[†]Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

overfitting and may not generalize to arbitrary shapes or scale to large crowds [31]. Optimization-based approaches have the ability to handle static shaped obstacles by explicitly formulating constraints using compact sets [41, 15] or building a prior grid map [28]. Recent works highlight the promise of combining learning with optimization to integrate these advantages, such as embedding differentiable model predictive control (MPC) into actor-critic learning [27, 35], and using learned dynamics inside sampling based MPC for agile adaptive control [37]. However, these methods mainly focus on single robot agile control, and their safety properties typically depend on the accuracy of learned costs or dynamics rather than explicit shape-aware safety terms.

To this end, this paper proposes `SRL-MPC`, a distributed shape-aware reinforcement learned Model Predictive Control (MPC) framework for robot navigation in crowds of robots and obstacles with explicitly modeled footprints. The key idea is to formulate a safe set based on geometric separation features (GSFs) as high-order control barrier function (HOCBF) constraints in the primal optimization problem for collision avoidance, while using RL to adjust MPC parameters based on the neighboring agent GSFs for task progress. After problem decomposition, the local MPC subproblem handles the HOCBF condition through a soft shape-aware residual penalty. In this way, RL does not replace the model-based planner; instead, it adapts the parameters of an explicit shape-aware HOCBF-MPC problem to balance collision avoidance and task progress. This approach has several advantages. First, unlike RL-based approaches that are sensitive to carefully designed reward functions and may suffer from limited generalization [14, 38], the proposed method learns parameter adaptation for an explicit shape-aware HOCBF-MPC problem rather than an unconstrained end-to-end policy. Second, the HOCBF-MPC parameters are adapted from neighboring GSFs, avoiding repeated manual weight tuning for different scenarios as required by many optimization-based approaches [22]. Third, the proposed method directly handles explicit convex geometric shape representations. For nonconvex footprints, the current implementation computes separation directions from the original geometry. Finally, following the problem decomposition technique in [16], the deployed local HOCBF-MPC subproblem is simple enough to solve in real time for each robot, resulting in a practical solution for dense dynamic crowd navigation.

To highlight the effectiveness of `SRL-MPC`, we evaluate it in highly randomized and dense scenarios consisting of multiple differential-drive robots, where the positions, goals, and shapes are all randomly generated. Additional evaluations cover out-of-distribution scenarios, variation across training seeds, and real-robot trials.

2 Related Work

Traditional Approaches. Traditional collision avoidance approaches often rely on geometric or optimization-based formulations. Recent VO-based methods such as Adaptive Optimal Collision Avoidance Driven by Opinion (`AVOCADO`) estimate an agent's cooperation level online through nonlinear opinion dynamics, improving collision avoidance in mixed crowds without communication [25]. MPC is a popular optimization framework that optimizes controls over a receding horizon with various constraints [40, 6]. To improve shape-aware collision avoidance, prior methods introduce disk primitives [42], polytopic velocity obstacles [19], sequential convex optimization [29], dual optimization-based collision avoidance (OBCA) constraints for convex sets [41], and accelerated optimization by problem decomposition and edge computation [15, 21]. These methods improve geometric fidelity, but their exact constraints can grow with the number of object surfaces. `SRL-MPC` follows this optimization line by using GSFs derived from a support function representation, a compact fixed-dimensional representation to encode convex objects without scaling with the number of surfaces.

Reinforcement Learning Approaches. RL approaches learn navigation policies from interaction data and have become an important category in socially aware and crowd-aware robot navigation [23, 8, 31]. Early socially aware deep reinforcement learning (DRL) methods learn collision-avoidance policies from local observations and social rewards [5, 7, 34]. Typical methods such as socially aware reinforcement learning (`SARL`) use attention mechanisms to encode human-robot interactions in a crowd-level representation [4]. More recent methods use richer sequence, occupancy-map, or transformer-based representations to reason about dynamic environments [33, 38]. These methods are flexible and adaptive in uncertain crowds, but safety and generalization remain difficult under unseen densities, agent behaviors, and body shapes. Safe RL methods add model-based lookahead or

shielding to reduce violations [1], but they do not directly provide explicit shape-aware safety terms. Consequently, few RL-based methods explicitly address arbitrarily shaped objects.

Hybrid Methods. Hybrid methods aim to combine the adaptability of learning techniques with the structure of model-based collision avoidance [26]. For example, reinforcement learning reciprocal velocity obstacle (`RL-RVO`) uses reciprocal velocity obstacle shaped rewards to guide distributed multi-robot policy learning [14]. Another line couples learned decision making with MPC controller execution in social navigation [3]. Recent Deep Residual MPC (`DR-MPC`) blends MPC path tracking with model-free DRL for real-world navigation and uses out-of-distribution detection with a heuristic safety check to reduce unsafe exploration [12]. More generally, learning-based MPC can learn dynamics models, costs, constraints, or terminal value approximations, and can also use MPC as a safety layer around RL policies [18, 39, 37, 27]. Unlike these methods, `SRL-MPC` does not learn the entire navigation policy or use optimization only as a shield for unsafe actions. Instead, RL adapts the HOCBF-MPC parameters from local GSFs, while the executed control is computed by the explicit shape-aware MPC optimization problem.

3 Problem Statement

Consider robot i navigating in a local crowd of N objects, including other robots, pedestrians, and static or dynamic obstacles. At each control cycle, robot i plans a state sequence $\mathcal{S}_i = \{\mathbf{s}_{i,0}, \dots, \mathbf{s}_{i,T}\}$ and a control sequence $\mathcal{U}_i = \{\mathbf{u}_{i,0}, \dots, \mathbf{u}_{i,T-1}\}$ over an MPC horizon T, where k denotes the prediction-step index. The state $\mathbf{s}_{i,k}$ contains the planar position $\mathbf{p}_{i,k} = [x_{i,k}, y_{i,k}]^\top$ and heading $\theta_{i,k}$, and $\mathbf{u}_{i,k}$ is the control input. The feasible set $\mathcal{F}_i$ collects the kinematic model, input bounds, input-increment bounds, and initial condition. The world-frame occupied set of robot i at step k is denoted by $\mathbb{Z}_i(\mathbf{s}_{i,k})$, which is obtained from a body-frame convex set $\mathbb{C}_i$ by state transformation. Detailed kinematic and geometric definitions are given in Appendix A.1.

Let $\mathcal{N}_i$ be the neighbor set considered by robot i, and let the bar notation denote nominal or predicted neighbor quantities, e.g., $\bar{\mathbf{s}}_{j,k}$ and $\bar{\mathbf{p}}_{j,k}$ for neighbor j. The navigation objective is to follow a reference path while maintaining a desired control profile. Let $\mathcal{S}_i^{\mathrm{ref}}$ and $\mathcal{U}_i^{\mathrm{ref}}$ denote the reference state and control sequences. Following the path-tracking objective used in [16], the cost function is

$$C_i(\mathcal{S}_i, \mathcal{U}_i) = w_p \|\mathbf{p}(\mathcal{S}_i) - \mathbf{p}(\mathcal{S}_i^{\mathrm{ref}})\|_2^2 + w_\theta \|\theta(\mathcal{S}_i) - \theta(\mathcal{S}_i^{\mathrm{ref}})\|_2^2 + w_u \|\mathcal{U}_i - \mathcal{U}_i^{\mathrm{ref}}\|_2^2, \tag{1}$$

where $\mathbf{p}(\mathcal{S}_i)$ and $\theta(\mathcal{S}_i)$ denote the stacked positions and headings extracted from $\mathcal{S}_i$, respectively, and w_p, w_θ, and w_u balance position tracking, heading tracking, and control effort. This cost encourages the robot to progress toward the goal while maintaining the desired control profile.

Collision avoidance is imposed through the minimum distance between occupied sets. For each neighbor $j \in \mathcal{N}_i$ and step k, define

$$D_{ij,k} = \min_{\mathbf{x}_i \in \mathbb{Z}_i(\mathbf{s}_{i,k}),\, \mathbf{x}_j \in \mathbb{Z}_j(\bar{\mathbf{s}}_{j,k})} \|\mathbf{x}_i - \mathbf{x}_j\|_2, \tag{2}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are world-frame points on the two occupied sets, and $D_{ij,k}$ is the minimum Euclidean distance between robot i and neighbor j. The required safety margin is denoted by d_{safe}. The resulting local planning problem is

$$\min_{\mathcal{S}_i, \mathcal{U}_i} C_i(\mathcal{S}_i, \mathcal{U}_i) \quad \text{s.t.} \quad (\mathcal{S}_i, \mathcal{U}_i) \in \mathcal{F}_i,\ D_{ij,k} \geq d_{\mathrm{safe}},\ j \in \mathcal{N}_i,\ k = 1, \dots, T. \tag{3}$$

where $(\mathcal{S}_i, \mathcal{U}_i)$ are the decision variables, $(\mathcal{S}_i, \mathcal{U}_i) \in \mathcal{F}_i$ enforces kinematic feasibility, and $D_{ij,k} \geq d_{\mathrm{safe}}$ enforces pairwise shape-aware separation along the horizon. This coupled problem is nonconvex and hard to solve directly. The next section introduces `SRL-MPC`, which separates geometry updates from local motion optimization and adapts the key MPC parameters online.

4 Method

The framework of the proposed method is illustrated in Figure 1. The method has two key components. First, it starts from the support function form of the pairwise minimum-distance problem and expresses the shape-aware safe set as degree-2 HOCBF constraints in the coupled primal problem. After problem decomposition, the GSFs are updated by a geometric solver, and the HOCBF condition is represented as a residual in the local MPC problem. Second, the reinforcement learning framework maps GSFs to MPC parameter updates, adapting tracking weights, desired speed weights, and the HOCBF safety distance to autonomously balance task completion efficiency and safety.

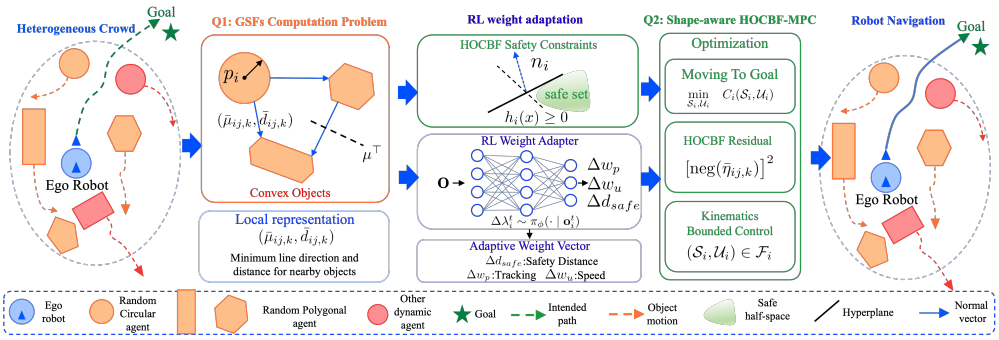


Figure 1: Overview of SRL-MPC. Key components: 1. A shape-aware HOCBF condition is formulated based on GSFs and works as a residual penalty in the optimization to improve safety; 2. the reinforcement learned policy adapts MPC parameters based on neighboring GSFs to handle dense scenarios.

4.1 Shape-Aware HOCBF Constraint

Few methods consider shape-aware HOCBF constraints because the exact conic constraints grow with the number of shape edges. Here we introduce the support function transformation [2] to formulate shape-aware HOCBF constraints based on fixed-dimensional GSFs. For a pair (i, j) and prediction step k, let $\bar{\mathbf{s}}_{j,k}$ be the nominal state of neighbor j. Under the strong-duality conditions used in [41, 15], the minimum-distance computation in (2), equivalently the body-frame program in (18), can be rewritten in dual form. To avoid the conic constraints, we further transform it into a support function form by:

$$D_{ij,k} = \max_{\|\mu\|_* \le 1} \Phi_{ij,k}(\mu), \quad \Phi_{ij,k}(\mu) = \mu^\top(\mathbf{p}_{i,k} - \bar{\mathbf{p}}_{j,k}) - \sigma_{\mathbb{C}_i}(-\mathbf{R}_{i,k}^\top \mu) - \sigma_{\mathbb{C}_j}(\bar{\mathbf{R}}_{j,k}^\top \mu), \tag{4}$$

where $\mu^\top$ represents the separating hyperplane between two convex occupied sets, μ is the minimum-distance direction, and $\|\cdot\|_*$ denotes the dual norm. We define the geometric separation feature as $\mathrm{GSF}_{ij,k} \triangleq (\mu_{ij,k}, D_{ij,k})$, where $\mu_{ij,k} \in \arg\max_{\|\mu\|_* \le 1} \Phi_{ij,k}(\mu)$ and $D_{ij,k}$ is the corresponding minimum distance. For a generic body-frame occupied set $\mathbb{C} \subset \mathbb{R}^2$ and a query direction $\boldsymbol{\xi} \in \mathbb{R}^2$, the support function $\sigma_{\mathbb{C}}(\boldsymbol{\xi}) = \sup\{\boldsymbol{\xi}^\top \mathbf{z} \mid \mathbf{z} \in \mathbb{C}\}$ gives the maximum projection of $\mathbb{C}$ along $\boldsymbol{\xi}$. Equation (4) therefore expresses the shape-aware minimum distance through the maximum projection induced by the separating hyperplane $\mu^\top$. Subtracting the required safety margin d_{safe} gives: $h_{ij,k}(\mu) = \Phi_{ij,k}(\mu) - d_{\text{safe}}$. For a selected $\mu_{ij,k}$, the collision avoidance condition at step k is written as $h_{ij,k}(\mu_{ij,k}) \ge 0$. Thus, the shape-aware safe set at prediction step k is

$$\mathcal{C}_{ij,k} = \{(\mathbf{s}_{i,k}, \bar{\mathbf{s}}_{j,k}) \mid H_{ij,k} \ge 0\}, \qquad H_{ij,k} \triangleq h_{ij,k}(\mu_{ij,k}). \tag{5}$$

A first-order discrete control barrier function (CBF) only imposes a one-step condition on the barrier value, which is weak for dense dynamic interactions with a high-speed profile. We therefore use a degree-2 discrete HOCBF [36]. For $\alpha_1, \alpha_2 \in (0, 1]$, the degree-2 HOCBF left-hand side is denoted as

$$\eta_{ij,k} = H_{ij,k+2} - (2 - \alpha_1 - \alpha_2) H_{ij,k+1} + (1 - \alpha_1)(1 - \alpha_2) H_{ij,k}. \tag{6}$$

In the primal problem, the HOCBF condition is imposed as the hard safety constraint $\eta_{ij,k} \ge 0$. With $\alpha_1 = \alpha_2 = \gamma$ and $\gamma \in (0, 1]$, this constraint becomes

$$H_{ij,k+2} - 2(1 - \gamma) H_{ij,k+1} + (1 - \gamma)^2 H_{ij,k} \ge 0, \qquad j \in \mathcal{N}_i^{\text{cbf}},\ k = 1, \dots, H_c. \tag{7}$$

where $\mathcal{N}_i^{\text{cbf}} \subseteq \mathcal{N}_i$ is the selected nearest-neighbor subset used for HOCBF terms, and $H_c \le T - 2$ is the HOCBF horizon.

Problem Decomposition: The minimum-distance line direction μ and the MPC trajectory $\mathbf{p}$ are coupled in the HOCBF constraint, leading to a bi-convex optimization problem. Following the block decomposition idea used in [15, 16], `SRL-MPC` separates the coupled problem into a GSF update subproblem Q_1 and a local HOCBF-MPC subproblem Q_2. The first block fixes nominal trajectories and computes nominal GSFs, while the second block fixes μ and computes the action sequence. After this decomposition, the HOCBF condition is handled through a soft residual penalty in Q_2. By iteratively solving Q_1 and Q_2, the geometric features and local trajectory are updated consistently, providing stronger optimization guidance than a static distance-margin penalty.

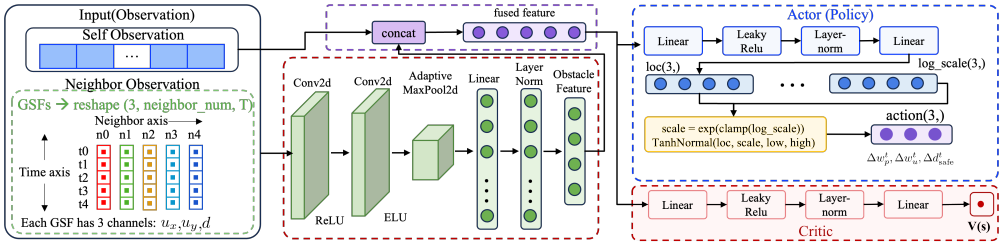


Figure 2: Neural network architecture. The GSFs are encoded by CNNs and then concatenated with the self-observation to form a fused feature, which is shared by the actor and critic.

Q_1: GSF computation subproblem. Given nominal trajectories $\bar{\mathcal{S}}_i$ and $\bar{\mathcal{S}}_j$, Q_1 computes the nominal GSFs between two transformed convex occupied sets. Instead of solving the optimization program, which is computationally expensive, we use a geometry-based shortest-line computation implemented by the Geometry Engine Open Source (GEOS) library [10], which is efficient for geometries:

$$\overline{\mathrm{GSF}}_{ij,k} \triangleq (\bar{\mu}_{ij,k}, \bar{d}_{ij,k}) = \mathrm{ShortestLine}\left(\mathbb{Z}_i(\bar{\mathbf{s}}_{i,k}), \mathbb{Z}_j(\bar{\mathbf{s}}_{j,k})\right), \tag{8}$$

where $\mathrm{ShortestLine}(\cdot)$ denotes the geometric shortest-line algorithm that returns the unit direction from the neighbor set to the ego set and the corresponding minimum distance. Specifically, for a circular set, $\bar{\mu}_{ij,k}$ can be obtained in closed form from the normalized center difference and the radius. Neighbors considered in the local MPC problem are ranked by the minimum distance over a short distance horizon, and only the closest K neighbors are retained in $\mathcal{N}_i$, where K is a user-specified neighbor budget.

Q_2: local HOCBF-MPC subproblem. Given the GSFs produced by Q_1, the local HOCBF-MPC subproblem Q_2 is formulated as

$$\begin{aligned} Q_2: \quad \min_{\mathcal{S}_i, \mathcal{U}_i} \quad & C_i(\mathcal{S}_i, \mathcal{U}_i) + \frac{\rho_{\text{obs}}}{2} \sum_{j \in \mathcal{N}_i^{\text{cbf}}} \sum_{k=1}^{H_c} [\mathrm{neg}(\bar{\eta}_{ij,k})]^2, \\ \text{s.t.} \quad & (\mathcal{S}_i, \mathcal{U}_i) \in \mathcal{F}_i. \end{aligned} \tag{9}$$

where $\rho_{\text{obs}} > 0$ is the penalty weight for HOCBF violation and is selected as a large value (e.g., 100). $\mathrm{neg}(x) = \max(0, -x)$ is the negative-part operator, and $\bar{\eta}_{ij,k}$ is the fixed-geometry counterpart of (6) after substituting the GSFs produced by Q_1. The penalty is zero when the decomposed high-order condition is satisfied and positive only when the predicted trajectory violates it. The soft penalty form avoids infeasibility that can occur when high-order conditions are imposed as hard local constraints in dense crowds, while keeping the local feasible set $\mathcal{F}_i$ unchanged. With the geometric terms fixed by Q_1 and the kinematics represented by affine linearization, Q_2 is a convex local HOCBF-MPC subproblem.

At each control cycle, the solver alternates these two subproblems for a small number of iterations. Nominal trajectories are first propagated from the current states. Then, for each robot, Q_1 updates the GSFs for the selected neighbors, Q_2 solves the local HOCBF-MPC problem, and the resulting trajectory becomes the nominal trajectory for the next iteration. The first control in the optimized sequence is applied to the controlled object.

4.2 Reinforcement Learned MPC

The solution of Q_2 depends strongly on the parameters in the MPC cost and safety terms, including w_p, w_θ, w_u, and d_{safe}. These weights and safety parameters determine how aggressively or conservatively a robot tracks the reference path (i.e., w_p), how strongly it penalizes control effort (i.e., w_u), and how much safety margin it requests from nearby agents (i.e., d_{safe}). These parameters usually require repeated manual tuning for each scenario. For example, in open space, larger w_u and w_p can guide the robot to track the reference path toward the goal quickly, while in dense scenarios, smaller w_p or d_{safe} may be needed to avoid collisions or getting stuck. This work proposes an RL-based parameter adaptation method to automatically tune these parameters, as illustrated in Figure 2, while retaining the explicit geometric and kinematic structure and generalizability of MPC. The learned policy observes local GSFs and current parameter values, outputs bounded parameter increments, and leaves

control synthesis to MPC. Because the GSF tensor stacks the closest neighbors over the horizon, the policy can adapt the MPC behavior to local crowd density and shape geometry.

Action space: The adaptive parameters are w_p, w_u, and d_{safe}, while w_θ is fixed to a small value, e.g., 0.01, to encourage robots to turn during collision avoidance. To make the parameters change smoothly over time, the policy outputs parameter increments rather than absolute parameter values. Thus, for robot i at time step t, the policy output is $\Delta\lambda_i^t = \begin{bmatrix}\Delta w_{p,i}^t & \Delta w_{u,i}^t & \Delta d_{\text{safe},i}^t\end{bmatrix}^\top$. The parameter vector is updated as

$$\lambda_i^{t+1} = \lambda_i^t + \Delta\lambda_i^t \in \Lambda, \quad \Lambda = [0.01, 1.0] \times [0.1, 10.0] \times [0.2, 1.0]. \tag{10}$$

The increment is bounded by $\Delta\lambda_i^t \in [-0.5, 0.5]\times[-0.5, 0.5]\times[-0.1, 0.1]$, where the third component is measured in meters. The absolute parameter values are clipped to the admissible set Λ, where $w_p \in [0.01, 1.0]$, $w_u \in [0.1, 10.0]$, and $d_{\text{safe}} \in [0.2, 1.0]$. A larger range is assigned to w_u to allow the policy to adjust the control effort strongly in dense scenarios.

Observation space: The robot observation $\mathbf{o}_i^t$ has two parts: the self-observation $\mathbf{o}_{i,\text{self}}^t$ and the neighbor observation $\mathbf{o}_{i,\text{neighbor}}^t$:

$$\mathbf{o}_{i,\text{self}}^t = \begin{bmatrix} v_i^t & w_{p,i}^t & w_{u,i}^t & d_{\text{safe},i}^t \end{bmatrix}^\top, \quad \mathbf{o}_{ij,k,\text{neighbor}}^t = \begin{bmatrix} \bar{\mu}_{ij,k,x}^t & \bar{\mu}_{ij,k,y}^t & \bar{d}_{ij,k}^t \end{bmatrix}^\top. \tag{11}$$

Here $\mathbf{o}_{ij,k,\text{neighbor}}^t$ stores GSFs, i.e., $(\bar{\mu}_{ij,k}^t, \bar{d}_{ij,k}^t)$ for neighbor j at horizon step k. Thus, $\mathbf{o}_{i,\text{neighbor}}^t \in \mathbb{R}^{3\times K\times T}$ stacks these geometric features for the K closest neighbors over the MPC horizon. The neighbors are selected by the minimum predicted distance over a short distance horizon, and the tensor is zero-padded when fewer than K neighbors are active. The self-observation includes the current MPC parameters $w_{p,i}^t$, $w_{u,i}^t$, $d_{\text{safe},i}^t$, and the forward speed v_i^t. Compared with general RL-based approaches [4, 14, 38], this observation space is compact and does not require explicit pose or goal-direction features. This compactness is possible because goal tracking and collision avoidance are handled by HOCBF-MPC; RL only learns how to adapt the MPC parameters to changing obstacle distributions.

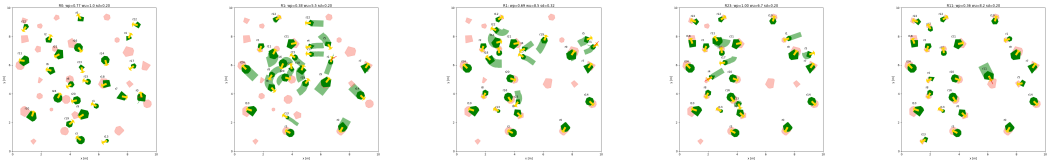
Neural network design. The neural network architecture is shown in Figure 2. The neighbor observation $\mathbf{o}_{i,\text{neighbor}}^t$ is treated as a three-channel image over the neighbor-horizon grid. The encoder applies two padded 3×3 convolutional layers, using a rectified linear unit (ReLU) after the first convolution and an exponential linear unit (ELU) after the second. This gives each output cell access to the full neighbor-horizon grid before pooling. An adaptive max-pooling layer keeps the strongest local interaction response, and a linear projection followed by layer normalization produces a fixed-size geometric feature.

The encoded geometric feature is concatenated with the 4-dimensional self-observation to form a shared feature. This shared encoder is used by both the actor and the critic. The actor outputs a 6-dimensional vector that is split into a 3-dimensional location vector and a 3-dimensional log-scale vector for the parameter increment $\Delta\lambda_i^t$. The resulting squashed Gaussian distribution samples bounded parameter increments. The critic uses the same shared encoder and a parallel linear head with LeakyReLU and layer normalization to output the scalar value estimate for proximal policy optimization (PPO) advantage computation. Layer normalization is used because the observation contains raw physical units, including velocity, MPC weights, and safety distance.

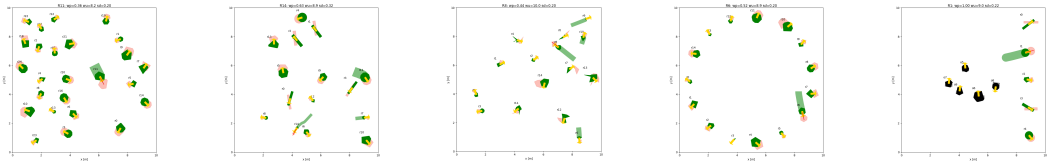
Reward design: The reward function is designed to align policy learning with task completion and safety. It uses a one-shot arrival bonus, a collision penalty, a small per-step time penalty, and a safety log-barrier penalty:

$$\begin{aligned} r_i^t &= r_{\text{arr}}\mathbf{1}\{i \text{ arrives}\} - r_{\text{col}}\mathbf{1}\{i \text{ collides}\} - r_{\text{safe}}\ell_{\text{safe},i}^t - r_{\text{step}}, \\ \ell_{\text{safe},i}^t &= \mathbf{1}\{d_{i,\text{near}}^t < d_{\text{safe},i}^t\}\,\text{clip}_{[0,1]}\left(\frac{\log\left(d_{\text{safe},i}^t / \max(d_{i,\text{near}}^t, \epsilon_d)\right)}{\log\left(d_{\text{safe}}^{\max}/d_{\text{safe}}^{\min}\right)}\right), \end{aligned} \tag{12}$$

Here $\mathbf{1}\{\cdot\}$ is a binary indicator, $d_{i,\text{near}}^t$ is the current minimum distance from robot i to its nearest neighbor, $d_{\text{safe},i}^t$ is the policy-adapted safety distance, and ϵ_d is a small numerical floor. The safety term is active only when the current minimum distance is smaller than the selected safety distance. The arrival bonus is assigned once when the robot reaches its goal and then parks at the goal, while



(a)



(b)

Figure 4: (a) Evaluation rollout with 25 randomly generated polygonal robots. (b) Random-polygon and out-of-distribution scenarios, including cross geometry, nonconvex union, circular polygon, and through traffic. Full rollouts are shown in Appendix A.4.

the collision penalty is assigned when a collision is detected. The collision penalty is larger than the arrival bonus, encouraging the policy to choose MPC parameters that reach the goal without relying on unsafe behaviors. The numerical reward constants are listed in Table 6 of Appendix A.5.

Training: The policy is trained with PPO [30] using decentralized execution and shared parameters. All robots share the same network parameters, and each robot reads the observation in (11) to generate its own MPC parameter increment. The training scenario is generated by IR-SIM [17] and shown in Figure 3: each episode samples 15 robots in a $10\,\mathrm{m} \times 10\,\mathrm{m}$ workspace with random start-goal pairs and convex polygon robot footprints. Each robot stops at its goal after arrival and then acts as a static obstacle for the other robots. A robot is reset when it collides or times out. Compared with other RL-based approaches trained with circular robots, sparse dynamic obstacles, or a fixed workspace, the training scenario is challenging because it combines randomized geometry, dense crowds, and both static and dynamic obstacles. The training process takes about 4 hours to achieve stable performance.

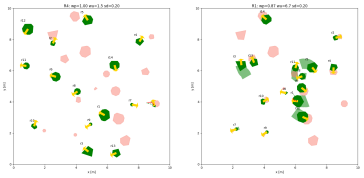


Figure 3: Training scenario examples. Each episode randomly samples start positions, goal positions, and convex polygon footprints; arrived robots park as static obstacles.

5 Experiments

Evaluation: We evaluate `SRL-MPC` in the same IR-SIM [17] randomized scenario family used for training, but with held-out random seeds, so no evaluation episode is reused during policy training. To test scalability and density robustness within this randomized shape-aware setting, the robot count is swept over $N \in \{10, 15, 20, 25\}$ to represent different crowd densities, while using the same model trained only in 15-robot scenarios. Figure 4 (a) shows a rollout with 25 randomly generated polygonal robots; Figure 4 (b) shows examples of out-of-distribution (OOD) scenarios. The OOD scenarios include cross geometry, nonconvex-union footprints, circular exchange, and through-traffic interactions, covering different geometric and crowd-flow patterns beyond the main random-polygon density sweep. Guided by `SRL-MPC`, each robot adapts its parameters to different neighbor geometries and reaches its goal successfully. Even in congested situations caused by parked robots, the ego robot adjusts its parameters to detour around the obstacles.

The main density-sweep evaluations use 100 episodes per robot count, a maximum episode length of 500 control steps, and the same random seeds across compared methods. An episode is counted as successful only when all robots reach their goals without collision or timeout. Navigation time and path length are averaged over arrived robots, while speed is averaged over active robot frames. All $\pm$ values denote standard deviations over the corresponding evaluated samples. The main simulation experiments are conducted on a MacBook Pro with an Apple M4 Pro CPU. The detailed controller, training, and evaluation parameters are summarized in Tables 5–7 of Appendix A.5.

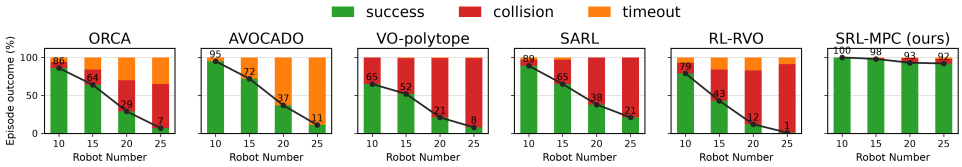


Figure 5: Episode outcomes for baseline methods in the random convex polygon scenario.

Table 1: Baseline comparison in the random-polygon setting

Robots	Method	Success↑	Collision↓	Timeout↓	Time (s)↓	Speed (m/s)↑	Path (m)↓
10	ORCA [32]	86.0	8.0	6.0	5.09 ± 2.58	0.878 ± 0.137	4.14 ± 2.25
	AVOCADO [25]	95.0	0.0	5.0	5.61 ± 2.63	0.785 ± 0.137	4.58 ± 2.28
	VO-polytope [19]	65.0	35.0	0.0	7.46 ± 3.91	0.536 ± 0.067	4.14 ± 2.25
	SARL [4]	89.0	8.0	3.0	8.90 ± 5.24	0.664 ± 0.164	5.78 ± 3.11
	RL-RVO [14]	79.0	14.0	7.0	6.99 ± 4.36	0.780 ± 0.136	5.42 ± 2.96
	SRL-MPC (ours)	**100.0**	0.0	0.0	6.11 ± 3.65	0.981 ± 0.091	6.02 ± 3.55
15	ORCA	64.0	20.0	16.0	5.22 ± 2.81	0.850 ± 0.151	3.94 ± 2.24
	AVOCADO	72.0	0.0	28.0	6.29 ± 4.41	0.727 ± 0.191	4.64 ± 2.42
	VO-polytope	52.0	47.0	1.0	7.16 ± 3.90	0.531 ± 0.065	3.94 ± 2.24
	SARL	65.0	32.0	3.0	8.38 ± 5.64	0.663 ± 0.166	5.44 ± 3.15
	RL-RVO	43.0	41.0	16.0	7.34 ± 4.86	0.734 ± 0.144	5.43 ± 3.31
	SRL-MPC (ours)	**98.0**	1.0	1.0	6.58 ± 4.00	0.970 ± 0.093	6.39 ± 3.80
20	ORCA	29.0	41.0	30.0	5.35 ± 3.35	0.816 ± 0.173	3.51 ± 2.20
	AVOCADO	37.0	0.0	63.0	6.69 ± 3.98	0.684 ± 0.209	4.76 ± 2.41
	VO-polytope	21.0	78.0	1.0	6.52 ± 3.99	0.511 ± 0.077	3.51 ± 2.20
	SARL	38.0	62.0	0.0	8.01 ± 5.05	0.653 ± 0.162	5.11 ± 2.88
	RL-RVO	12.0	71.0	17.0	7.53 ± 5.79	0.685 ± 0.155	5.32 ± 3.59
	SRL-MPC (ours)	**93.0**	6.0	1.0	7.26 ± 4.44	0.960 ± 0.080	6.93 ± 4.12
25	ORCA	7.0	58.0	35.0	5.38 ± 3.49	0.780 ± 0.191	2.84 ± 2.00
	AVOCADO	11.0	0.0	89.0	7.64 ± 5.51	0.638 ± 0.228	4.91 ± 2.52
	VO-polytope	8.0	91.0	1.0	5.38 ± 3.68	0.496 ± 0.086	2.84 ± 2.00
	SARL	21.0	79.0	0.0	7.38 ± 4.70	0.639 ± 0.168	4.75 ± 2.81
	RL-RVO	1.0	90.0	9.0	6.71 ± 5.95	0.657 ± 0.162	4.74 ± 4.03
	SRL-MPC (ours)	**92.0**	6.0	2.0	7.94 ± 5.00	0.943 ± 0.093	7.41 ± 4.41

Table 2: Baseline comparison in out-of-distribution settings. Entries report success↑/collision↓/timeout↓ (%); L denotes the number of outer iterations.

Method	Nonconvex union	Cross geometry	Circular exchange	Through traffic	Heterogeneous dynamics
ORCA [32]	83/10/7	71/5/24	91/7/2	92/1/7	88/3/9
AVOCADO [25]	90/2/8	61/0/39	97/0/3	**95**/1/4	85/0/15
VO-polytope [19]	7/93/0	6/94/0	1/99/0	44/55/1	27/73/0
SARL [4]	72/4/24	44/4/52	**100**/0/0	44/4/52	49/0/51
RL-RVO [14]	45/46/9	17/78/5	9/91/0	37/60/3	66/30/4
SRL-MPC (ours, $L = 1$)	**99**/1/0	76/22/2	56/44/0	85/15/0	**97**/1/2
SRL-MPC (ours, $L = 2$)	96/2/2	**87**/13/0	83/17/0	77/23/0	96/2/2

Comparison with Baselines: We compare SRL-MPC with five representative baselines: ORCA, a classic reciprocal velocity obstacle method [32]; AVOCADO, an adaptive optimal collision avoidance method with opinion dynamics [25]; VO-polytope, a polygon-precise velocity obstacle method [19]; SARL, a socially aware reinforcement learning policy that uses attention to encode crowd interactions [4]; and RL-RVO, a pretrained reinforcement learning policy based on reciprocal velocity obstacle features [14]. These baselines cover reactive, geometry-aware, learning-based, and adaptive collision avoidance methods commonly used in crowd navigation. Figure 6 shows the real-world experiment setup.

Random-polygon setting. Table 1 shows that SRL-MPC achieves the highest success rate across all robot counts. The gap becomes larger in dense scenes: with 25 robots, SRL-MPC reaches 92.0% success, while ORCA, AVOCADO, VO-polytope, SARL, and RL-RVO achieve 7.0%, 11.0%, 8.0%, 21.0%, and 1.0%, respectively. The improvement over the strongest external baseline is 55.0 percentage points at 20 robots and 71.0 percentage points at 25 robots. ORCA often has the shortest navigation time and path length among arrived robots, but these averages exclude many failed robots in dense scenes. AVOCADO avoids collisions but produces many timeout episodes in dense settings. Although VO-polytope models polygonal geometry more explicitly than circular VO methods, it remains a reactive pairwise velocity-obstacle method. In dense multi-robot scenes, the polygonal

Table 3: Ablation study in the random convex polygon scenario.

Robots	Method	Success↑	Collision↓	Timeout↓	Time (s)↓	Speed (m/s)↑	Path (m)↓
10	`dist_mpc` (w/o HOCBF+RL)	77.0	21.0	2.0	4.99 ± 2.49	0.997 ± 0.107	5.00 ± 2.51
10	`manual_mpc` (w/o RL)	96.0	1.0	3.0	6.69 ± 5.32	0.983 ± 0.091	6.60 ± 5.24
10	`rule_mpc` (rule, w/o RL)	94.0	0.0	6.0	6.87 ± 5.55	0.980 ± 0.094	6.76 ± 5.43
10	`SRL-MPC` (ours, with HOCBF+RL)	**100.0**	0.0	0.0	6.11 ± 3.65	0.981 ± 0.091	6.02 ± 3.55
15	`dist_mpc` (w/o HOCBF+RL)	51.0	46.0	3.0	4.81 ± 2.73	0.997 ± 0.106	4.81 ± 2.74
15	`manual_mpc` (w/o RL)	93.0	2.0	5.0	6.85 ± 4.74	0.969 ± 0.101	6.64 ± 4.53
15	`rule_mpc` (rule, w/o RL)	92.0	1.0	7.0	7.21 ± 5.27	0.965 ± 0.098	6.92 ± 4.90
15	`SRL-MPC` (ours, with HOCBF+RL)	**98.0**	1.0	1.0	6.58 ± 4.00	0.970 ± 0.093	6.39 ± 3.80
20	`dist_mpc` (w/o HOCBF+RL)	15.0	84.0	1.0	3.91 ± 2.67	1.003 ± 0.097	3.90 ± 2.60
20	`manual_mpc` (w/o RL)	75.0	5.0	20.0	7.43 ± 5.12	0.959 ± 0.102	7.13 ± 4.85
20	`rule_mpc` (rule, w/o RL)	85.0	1.0	14.0	7.93 ± 5.30	0.950 ± 0.090	7.43 ± 4.77
20	`SRL-MPC` (ours, with HOCBF+RL)	**93.0**	6.0	1.0	7.26 ± 4.44	0.960 ± 0.080	6.93 ± 4.12
25	`dist_mpc` (w/o HOCBF+RL)	7.0	92.0	1.0	3.57 ± 2.70	1.002 ± 0.106	3.55 ± 2.68
25	`manual_mpc` (w/o RL)	68.0	13.0	19.0	7.72 ± 5.05	0.945 ± 0.110	7.27 ± 4.62
25	`rule_mpc` (rule, w/o RL)	75.0	2.0	23.0	8.67 ± 5.97	0.931 ± 0.103	7.94 ± 5.22
25	`SRL-MPC` (ours, with HOCBF+RL)	**92.0**	6.0	2.0	7.94 ± 5.00	0.943 ± 0.093	7.41 ± 4.41

velocity cones become tight and overlapping, and the method lacks receding-horizon optimization or adaptive parameter tuning to resolve multi-way conflicts; therefore, its failure mode becomes collision-heavy as density increases. Together, Table 1 and Figure 5 show that `SRL-MPC` improves the task-completion and safety tradeoff, rather than only changing the speed profile of successful trajectories.

Out-of-distribution setting. Without retraining, we compare the same methods in five OOD settings over 100 episodes per method and setting (Table 2). With one iteration, `SRL-MPC` success is $99/76/56/85/97\%$ for nonconvex union, cross geometry, circular exchange, through traffic, and heterogeneous dynamics, respectively. Circular-exchange success increases to 83% with two iterations, versus 100% for `SARL`. The $22/44/15\%$ collision rates in cross, circular, and traffic settings indicate sensitivity to structured interactions.

Ablation Study: To validate the functionality of the proposed components, we also run an ablation study with `dist_mpc`, `manual_mpc`, `rule_mpc`, and `SRL-MPC`. `dist_mpc` uses neither RL nor HOCBF; it replaces the HOCBF term with a static distance-margin penalty over the horizon, $J_{\text{dist}} = (\rho_{\text{obs}}/2) \sum_{j \in \mathcal{N}_i} \sum_{k=1}^{T} [\text{neg}(\bar{H}_{ij,k|k})]^2$, where $\bar{H}_{ij,k|k}$ is the fixed-geometry distance barrier obtained from the GSFs at step k. `manual_mpc` uses HOCBF without RL by solving the same HOCBF-MPC problem with fixed handcrafted MPC parameters. `rule_mpc` also uses HOCBF without RL, but replaces learned adaptation with a hand coded stuck rule. It uses the default parameters $(w_p, w_u, d_{\text{safe}}) = (0.01, 10.0, 0.30)$ and switches to $(1.0, 0.1, 0.20)$ when the robot satisfies $|v_i| < 0.10\,\text{m/s}$ for 10 consecutive control steps. `SRL-MPC` uses both HOCBF and RL based continuous parameter adaptation. Table 3 shows the superior performance of `SRL-MPC` over the ablation baselines. `manual_mpc` is competitive at low and moderate densities, but drops to 75.0% and 68.0% success at 20 and 25 robots, whereas `SRL-MPC` reaches 93.0% and 92.0%. `rule_mpc` improves over `manual_mpc` at high density by using a binary stuck rule: at 20 and 25 robots, success rates increase from 75.0% to 85.0% and from 68.0% to 75.0%, respectively. However, `rule_mpc` still depends on hand tuned thresholds and only switches between two parameter settings. Its wider default safety distance also makes it prone to timeouts, with 23.0% timeout at 25 robots. In contrast, `SRL-MPC` improves over `rule_mpc` by 6.0, 6.0, 8.0, and 17.0 percentage points for 10, 15, 20, and 25 robots, respectively, by adapting parameters continuously from local GSFs. At 25 robots, `rule_mpc` remains collision conservative but timeout heavy, whereas `SRL-MPC` reaches 92.0% success by reducing timeout to 2.0% while keeping collision to 6.0%. `dist_mpc` is fast among the robots that arrive, but its collision rate increases to 84.0% at 20 robots and 92.0% at 25 robots, confirming that the static distance margin penalty is insufficient as the primary safety mechanism in dense polygonal crowds. Unlike the static distance margin penalty, the HOCBF residual couples barrier values across consecutive predicted steps, so it penalizes not only instantaneous distance violations but also unsafe trends along the horizon. This provides more informative optimization guidance in dense interactions. The comparison isolates the benefits of the HOCBF residual, rule based adaptation, and learned continuous parameter adaptation in sequence.

Robustness Analysis: We further evaluate `SRL-MPC` under perception noise, action delay, and parameter delay in the $N = 15$ random polygon setting, using the same scenario, evaluation seed, and max-step setting as the main evaluation. The success rate remains 100.0% with Gaussian neighbor-

Table 4: Robustness analysis under perception noise and delay perturbations for $N = 15$.

Perception noise				Action delay				Parameter delay			
σ (m)	Succ.	Coll.	T.out	K (step)	Succ.	Coll.	T.out	K (step)	Succ.	Coll.	T.out
0.00	98.0	1.0	1.0	0	98.0	1.0	1.0	0	98.0	1.0	1.0
0.02	100.0	0.0	0.0	1	99.0	1.0	0.0	1	98.0	2.0	0.0
0.05	83.0	17.0	0.0	2	82.0	17.0	1.0	2	100.0	0.0	0.0
0.10	40.0	60.0	0.0	3	2.0	98.0	0.0	3	99.0	1.0	0.0
0.20	1.0	99.0	0.0	5	0.0	100.0	0.0	5	100.0	0.0	0.0

position noise of $\sigma = 0.02\,\mathrm{m}$ and 83.0% at $\sigma = 0.05\,\mathrm{m}$, and reaches 99.0% with one-step action delay ($0.1\,\mathrm{s}$). The method is also insensitive to delayed RL parameter updates: with a five-step parameter delay ($0.5\,\mathrm{s}$), success remains 100.0%. Larger perception noise or action delay causes a clear collision increase, indicating that accurate short-horizon state estimation and low-latency actuation remain important. The detailed sweep is reported in Table 4. We also deploy `SRL-MPC` on real-world robot platforms to validate practical effectiveness and real-time execution, as shown in Figure 6 and the supplementary video.

Additional evaluations. *(i) Outer-iteration ablation* (Appendix A.8). In the additional 20-robot sweep, one to four iterations increase `SRL-MPC` success from 94% to 100% and reduce collisions from 4% to 0%, while mean controller time increases from 9.500 to 36.703 ms per robot and step. At four iterations, `manual_mpc` also has zero observed collisions but 71% timeouts. The appendix reports the complete sweeps and separate residual diagnostics; performance is not monotonic in iteration count. *(ii) Training-seed variation* (Appendix A.9). Three additional PPO training seeds yield $91.0 \pm 2.6\%$ and $89.7 \pm 0.6\%$ success at 20 and 25 robots, respectively. Table 10 reports the dense and OOD results, with sample standard deviations across trained policies. *(iii) Hardware evaluation* (Appendix A.10). Six robots are evaluated over 20 paired trials per method (Table 11). `SRL-MPC` achieves 85% success and 15% collisions, versus 60% and 40% for `ORCA`. The robots share the same physical chassis; different footprint sizes are configured in the controllers. Navigation time and speed are computed over successful trials. *(iv) Computational cost* (Appendix A.11). We report GSF backend timings, a module breakdown, and comparisons with RDA and an online OBCA adapter.

6 Conclusion

This paper presents `SRL-MPC`, a shape-aware reinforcement learned MPC framework for collision avoidance in crowded dynamic environments with multiple robots and obstacles. It represents geometric constraints through GSFs based on support function transformation, formulates degree-2 HOCBF constraints in the coupled primal problem, and decomposes online planning into a GSF update and a local HOCBF-MPC subproblem where the decomposed HOCBF residual is softly penalized. Reinforcement learning adapts MPC parameters from neighboring shape-aware geometric features, while the executed control remains the solution of an explicit MPC optimization problem. Experiments in random polygon scenarios with density sweeps show that `SRL-MPC` achieves the highest success rate among representative baselines, especially in dense scenes where the baselines degrade sharply. The HOCBF residual improves empirical collision avoidance, while learned parameter adaptation primarily reduces timeouts and improves task progress. Future work includes uncertainty-aware neighbor prediction, richer nonconvex body decompositions, and real-world deployment with onboard sensing and computation.

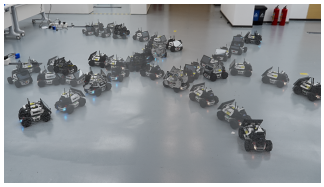


Figure 6: Real-world platform deployment.

Acknowledgments and Disclosure of Funding

This work is supported by the National Natural Science Foundation of China (No. 62422606), Hong Kong Research Grant Council General Research Fund (No. 17213925), Shenzhen Loop Area Institute Fund (No. FPF10120250006), and HKU Shanghai Intelligent Computing Research Center. This work is also partially supported by the National Natural Science Foundation of China (Grant No. 62371444), and the Shenzhen Science and Technology Program (Grant No. RCYX20231211090206005, JCYJ20241202124934046).

References

[1] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, pages 2669–2678, 2018. doi: 10.1609/aaai.v32i1.11797.

[2] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[3] Bruno Brito, Michael Everett, Jonathan P. How, and Javier Alonso-Mora. Where to go next: Learning a subgoal recommendation policy for navigation in dynamic environments. *IEEE Robotics and Automation Letters*, 6(3):4616–4623, 2021.

[4] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 international conference on robotics and automation (ICRA)*, pages 6015–6022. IEEE, 2019.

[5] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017.

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

[7] Michael Everett, Yu Fan Chen, and Jonathan P. How. Collision avoidance in pedestrian-rich environments with deep reinforcement learning. *IEEE Access*, 9:10357–10377, 2021. doi: 10.1109/ACCESS.2021.3050338.

[8] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research*, 39(7):856–892, 2020.

[9] Anthony Francis, Claudia Perez-D'Arpino, Chengshu Li, Fei Xia, Alexandre Alahi, et al. Principles and guidelines for evaluating social robot navigation algorithms. *ACM Transactions on Human-Robot Interaction*, 14(2):1–65, 2025. doi: 10.1145/3700599.

[10] GEOS contributors. *GEOS computational geometry library*. Open Source Geospatial Foundation, 2025. URL `https://libgeos.org/`.

[11] Ke Guo, Dawei Wang, Tingxiang Fan, and Jia Pan. Vr-orca: Variable responsibility optimal reciprocal collision avoidance. *IEEE Robotics and Automation Letters*, 6(3):4520–4527, 2021.

[12] James R. Han, Hugues Thomas, Jian Zhang, Nicholas Rhinehart, and Timothy D. Barfoot. DR-MPC: Deep residual model predictive control for real-world social navigation. *IEEE Robotics and Automation Letters*, 10(4):4029–4036, 2025. doi: 10.1109/LRA.2025.3546106.

[13] Ruihua Han, Shengduo Chen, and Qi Hao. Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 448–454. IEEE, 2020.

[14] Ruihua Han, Shengduo Chen, Shuaijun Wang, Zeqing Zhang, Rui Gao, Qi Hao, and Jia Pan. Reinforcement learned distributed multi-robot navigation with reciprocal velocity obstacle shaped rewards. *IEEE Robotics and Automation Letters*, 7(3):5896–5903, 2022. doi: 10.1109/LRA.2022.3161699.

[15] Ruihua Han, Shuai Wang, Shuaijun Wang, Zeqing Zhang, Qianru Zhang, Yonina C Eldar, Qi Hao, and Jia Pan. Rda: An accelerated collision free motion planner for autonomous navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 8(3):1715–1722, 2023. doi: 10.1109/LRA.2023.3242138.

[16] Ruihua Han, Shuai Wang, Shuaijun Wang, Zeqing Zhang, Jianjun Chen, Shijie Lin, Chengyang Li, Chengzhong Xu, Yonina C Eldar, Qi Hao, et al. NeuPAN: Direct point robot navigation with end-to-end model-based learning. *IEEE Transactions on Robotics*, 41:2804–2824, 2025. doi: 10.1109/TRO.2025.3554252.

[17] Ruihua Han, Shuai Wang, Chengyang Li, Rui Gao, Xinyi Wang, Zhe Liu, Guoliang Li, Yupu Lu, Qi Hao, Jia Pan, and Hengshuang Zhao. IR-SIM: A lightweight declarative simulator for navigation learning and benchmarking. *arXiv preprint arXiv:2606.08729*, 2026. doi: 10.48550/arXiv.2606.08729. URL `https://arxiv.org/abs/2606.08729`.

[18] Lukas Hewing, Kim P. Wabersich, Marcel Menner, and Melanie N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020. doi: 10.1146/annurev-control-090419-075625.

[19] Jihao Huang, Jun Zeng, Xuemin Chi, Koushil Sreenath, Zhitao Liu, and Hongye Su. Velocity obstacle for polytopic collision avoidance for distributed multi-robot systems. *IEEE Robotics and Automation Letters*, 8(6):3502–3509, 2023.

[20] Taekyung Kim, Robin Inho Kee, and Dimitra Panagou. Learning to refine input constrained control barrier functions via uncertainty-aware online parameter adaptation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3868–3875, 2025. doi: 10.1109/ICRA55743.2025.11128840.

[21] Guoliang Li, Ruihua Han, Shuai Wang, Fei Gao, Yonina C Eldar, and Chengzhong Xu. Edge accelerated robot navigation with collaborative motion planning. *IEEE/ASME Transactions on Mechatronics*, 30(2):1166–1178, 2025. doi: 10.1109/TMECH.2024.3419436.

[22] Yisheng Li, Longji Yin, Yixi Cai, Jianheng Liu, Fangcheng Zhu, Mingpu Ma, Siqi Liang, Haotian Li, and Fu Zhang. Efficient swept volume-based trajectory generation for arbitrary-shaped ground robot navigation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2076–2083. IEEE, 2025. doi: 10.1109/IROS60139.2025.11247085.

[23] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6252–6259. IEEE, 2018.

[24] Wenhao Luo, Wen Sun, and Ashish Kapoor. Multi-robot collision avoidance under uncertainty with probabilistic safety barrier certificates. In *Advances in Neural Information Processing Systems*, volume 33, pages 372–383, 2020.

[25] Diego Martinez-Baselga, Eduardo Sebastián, Eduardo Montijano, Luis Riazuelo, Carlos Sagüés, and Luis Montano. AVOCADO: Adaptive optimal collision avoidance driven by opinion. *IEEE Transactions on Robotics*, 41:2495–2511, 2025. doi: 10.1109/TRO.2025.3552350.

[26] Jianmin Qin, Jiahu Qin, Jiaxin Qiu, Qingchen Liu, Man Li, and Qichao Ma. SRL-ORCA: A socially aware multi-agent mapless navigation algorithm in complex dynamic scenes. *IEEE Robotics and Automation Letters*, 9(1):143–150, 2024. doi: 10.1109/LRA.2023.3331621.

[27] Angel Romero, Elie Aljalbout, Yunlong Song, and Davide Scaramuzza. Actor–critic model predictive control: Differentiable optimization meets reinforcement learning for agile flight. *IEEE Transactions on Robotics*, 42:673–692, 2026. doi: 10.1109/TRO.2025.3644945.

[28] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Kinodynamic trajectory optimization and control for car-like robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5681–5686. IEEE, 2017.

[29] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[31] Phani Teja Singamaneni, Pilar Bachiller-Burgos, Luis J. Manso, Anais Garrell, Alberto Sanfeliu, Anne Spalanzani, and Rachid Alami. A survey on socially aware robot navigation: Taxonomy and future challenges. *The International Journal of Robotics Research*, 43(10):1533–1572, 2024. doi: 10.1177/02783649241230562.

[32] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics Research: The 14th International Symposium ISRR*, pages 3–19. Springer, 2011.

[33] Haitong Wang, Aaron Hao Tan, and Goldie Nejat. Navformer: A transformer architecture for robot target-driven navigation in unknown and dynamic environments. *IEEE Robotics and Automation Letters*, 9(8):6808–6815, 2024.

[34] Shuaijun Wang, Rui Gao, Ruihua Han, Shengduo Chen, Chengyang Li, and Qi Hao. Adaptive environment modeling based reinforcement learning for collision avoidance in complex scenes. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9011–9018. IEEE, 2022.

[35] Xinyi Wang, Taekyung Kim, Bardh Hoxha, Georgios Fainekos, and Dimitra Panagou. Diffcvar: Reinforcement learning for risk adaptation via differentiable cvar barrier functions. *IEEE Robotics and Automation Letters*, 2026.

[36] Wei Xiao and Calin Belta. High-order control barrier functions. *IEEE Transactions on Automatic Control*, 67(7):3655–3662, 2022. doi: 10.1109/TAC.2021.3105491.

[37] Wenli Xiao, Haoru Xue, Tony Tao, Dvij Kalaria, John M Dolan, and Guanya Shi. AnyCar to anywhere: Learning universal dynamics model for agile and adaptive mobility. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8819–8825. IEEE, 2025. doi: 10.1109/ICRA55743.2025.11128396.

[38] Zhefan Xu, Xinming Han, Haoyu Shen, Hanyu Jin, and Kenji Shimada. Navrl: Learning safe flight in dynamic environments. *IEEE Robotics and Automation Letters*, 10(4):3668–3675, 2025. doi: 10.1109/LRA.2025.3546069.

[39] Mario Zanon and Sébastien Gros. Safe reinforcement learning using robust mpc. *IEEE Transactions on Automatic Control*, 66(8):3638–3652, 2020.

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

[41] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology*, 29(3):972–983, 2020.

[42] Julius Ziegler and Christoph Stiller. Fast collision checking for intelligent vehicle motion planning. In *2010 IEEE intelligent vehicles symposium*, pages 518–522. IEEE, 2010.

A Technical appendices and supplementary material

A.1 Kinematic and Geometric Model

This appendix provides the detailed definitions of the feasible set $\mathcal{F}_i$ and the occupied set $\mathbb{Z}_i(\mathbf{s}_{i,k})$ used in Section 3. The state and control satisfy

$$\mathbf{s}_{i,k+1} = f_i(\mathbf{s}_{i,k}, \mathbf{u}_{i,k}), \quad k = 0, \dots, T-1, \tag{13}$$

where $f_i : \mathcal{X}_i \times \mathcal{V}_i \to \mathbb{R}^n$ is the discrete-time robot dynamics, $\mathcal{X}_i$ is the state space, and $\mathcal{V}_i$ is the admissible input set. In planar navigation, $n = 3$ and $\mathbf{s}_{i,k} = [x_{i,k}, y_{i,k}, \theta_{i,k}]^\top$. The implementation uses a differential-drive model with $\mathbf{u}_{i,k} = [v_{i,k}, \omega_{i,k}]^\top$ and sampling time Δt:

$$x_{i,k+1} = x_{i,k} + \Delta t\, v_{i,k} \cos\theta_{i,k}, \quad y_{i,k+1} = y_{i,k} + \Delta t\, v_{i,k} \sin\theta_{i,k}, \quad \theta_{i,k+1} = \theta_{i,k} + \Delta t\, \omega_{i,k}. \tag{14}$$

The nonlinear kinematics are linearized around the nominal trajectory as in [15]. The input and input increment satisfy $\mathbf{u}_{i,\min} \le \mathbf{u}_{i,k} \le \mathbf{u}_{i,\max}$ and $\Delta\mathbf{u}_{i,\min} \le \Delta\mathbf{u}_{i,k} \le \Delta\mathbf{u}_{i,\max}$ for $k = 0, \dots, T-1$, where $\Delta\mathbf{u}_{i,k} = \mathbf{u}_{i,k} - \mathbf{u}_{i,k-1}$ and $\mathbf{u}_{i,-1}$ is the previously applied control. Thus, $\mathcal{F}_i$ is the set of all $(\mathcal{S}_i, \mathcal{U}_i)$ satisfying (13), the input constraints, and the initial condition $\mathbf{s}_{i,0} = \mathbf{s}_i^{\mathrm{cur}}$.

Following the geometric convention in [41, 16], the convex body-frame occupied set of object i and its world-frame transformation at prediction step k are

$$\mathbb{C}_i = \{\, \mathbf{z} \in \mathbb{R}^2 \mid \mathbf{G}_i\mathbf{z} \preceq_{\mathcal{K}_i} \mathbf{h}_i \,\}, \quad \mathbb{Z}_i(\mathbf{s}_{i,k}) = \{\, \mathbf{R}_{i,k}\mathbf{z} + \mathbf{p}_{i,k} \mid \mathbf{z} \in \mathbb{C}_i \,\}. \tag{15}$$

Here, $\mathbf{z}$ is a body-frame point, $\mathbf{G}_i$ and $\mathbf{h}_i$ define the conic inequalities, $\mathcal{K}_i$ is a proper cone, and $\preceq_{\mathcal{K}_i}$ denotes the partial order induced by $\mathcal{K}_i$. The matrix $\mathbf{R}_{i,k} \triangleq \mathbf{R}(\theta_{i,k}) \in SO(2)$ is the planar rotation matrix, where

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{16}$$

For a nominal neighbor state $\bar{\mathbf{s}}_{j,k}$, the corresponding position and rotation are denoted by $\bar{\mathbf{p}}_{j,k}$ and $\bar{\mathbf{R}}_{j,k}$. The support function used in (4) is defined as

$$\sigma_{\mathbb{C}}(\boldsymbol{\xi}) = \sup\{\, \boldsymbol{\xi}^\top\mathbf{z} \mid \mathbf{z} \in \mathbb{C} \,\}, \quad \sigma_{\mathbb{C}_{\mathrm{cir}}}(\boldsymbol{\xi}) = r\|\boldsymbol{\xi}\|_2, \quad \sigma_{\mathbb{C}_{\mathrm{poly}}}(\boldsymbol{\xi}) = \max_{a=1,\dots,n_v} (\mathbf{v}^a)^\top\boldsymbol{\xi}, \tag{17}$$

where $\mathbb{C}_{\mathrm{cir}}$ is a body-centered circle with radius r, and $\mathbb{C}_{\mathrm{poly}}$ is a polygon with body-frame vertices $\{\mathbf{v}^a\}_{a=1}^{n_v}$. Substituting (15) into (2) gives the equivalent body-frame distance program

$$\begin{aligned} D_{ij,k} = \min_{\mathbf{z}_i, \mathbf{z}_j} \; & \left\| \mathbf{R}_{i,k}\mathbf{z}_i + \mathbf{p}_{i,k} - \left(\bar{\mathbf{R}}_{j,k}\mathbf{z}_j + \bar{\mathbf{p}}_{j,k}\right) \right\|_2 \\ \text{s.t.} \quad & \mathbf{G}_i\mathbf{z}_i \preceq_{\mathcal{K}_i} \mathbf{h}_i, \quad \mathbf{G}_j\mathbf{z}_j \preceq_{\mathcal{K}_j} \mathbf{h}_j. \end{aligned} \tag{18}$$

A.2 Limitations

The main limitation is the computational cost of solving the local HOCBF-MPC subproblem. As shown in Table 8, `SRL-MPC` is heavier than lightweight reactive baselines because each control step solves a convex MPC problem after the geometric feature update. Nevertheless, the measured mean per-robot controller time remains at the $10\,\mathrm{ms}$ level in the tested dense scenarios: 5.79, 6.73, 9.37, and $10.34\,\mathrm{ms}$ for 10, 15, 20, and 25 robots, respectively, supporting real-time per-robot control in the platform experiment shown in Figure 6.

A.3 Broad Impact

This work aims to improve the safety and reliability of autonomous robot navigation in shared spaces, including warehouses, service-robot environments, and heterogeneous robot fleets. By keeping the executed control inside an explicit MPC optimization problem and using RL to adapt interpretable parameters, the framework may reduce the risk of opaque end-to-end policy behavior in dense navigation tasks. Potential negative impacts mainly come from premature deployment: inaccurate state estimation, model mismatch, computation delays, or unmodeled pedestrian behavior could still lead to unsafe motion.

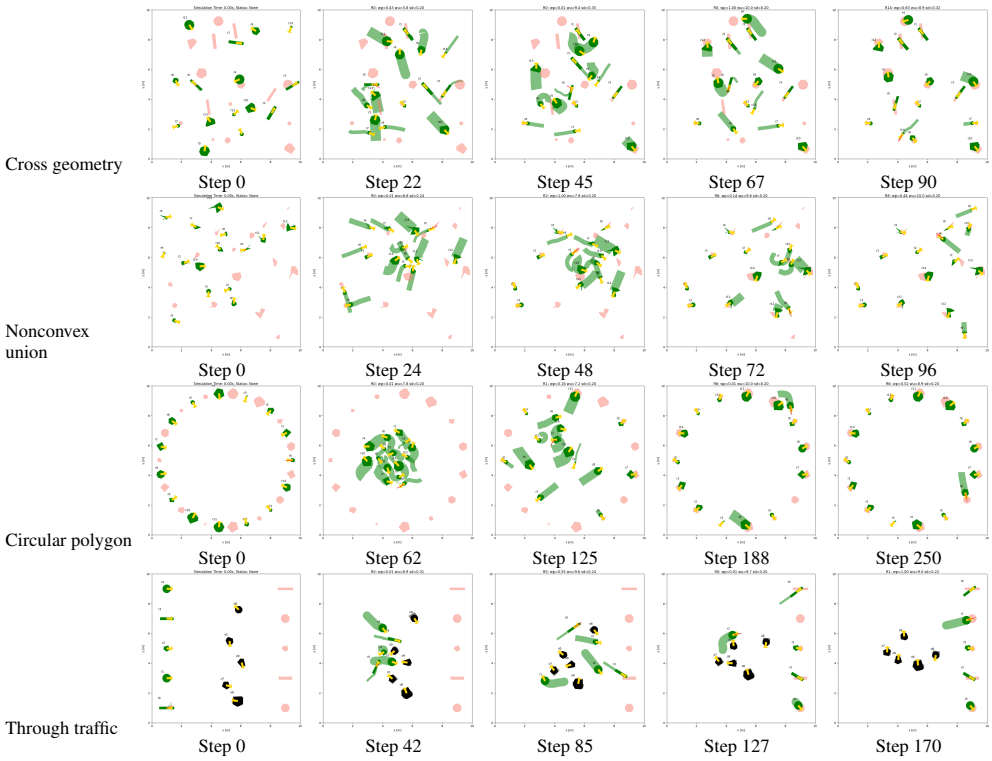


Figure 7: Additional challenging scenario rollouts. Rows show cross geometry, nonconvex union, circular polygon, and through traffic scenarios sampled from held-out seeds.

A.4 Additional Challenging Scenarios

Figure 7 shows additional held-out scenarios, including cross geometry, nonconvex union, circular polygon, and through traffic layouts. The snapshots are sampled from one episode per scenario from the beginning to the final step, illustrating that `SRL-MPC` can guide polygonal robots through different geometry layouts while adapting the MPC parameters during the rollout. Section A.4.1 further discusses a reactive-pair example.

A.4.1 Reactive Parameter Adaptation

Figure 8 visualizes a close interaction between two robots and the corresponding learned parameter changes. As the neighboring geometry changes during the interaction, the policy reacts by updating w_p, w_u, and d_{safe} online. This example validates that the learned adaptation is not a fixed parameter schedule, but responds to the current GSFs induced by nearby robot shapes and distances.

A.5 Implementation and Experiment Parameters

This subsection summarizes the main implementation settings used for the reported experiments. Table 5 lists the controller and scenario parameters, Table 6 lists the reinforcement learning and neural-network parameters, and Table 7 lists the evaluation and baseline settings. The notation follows the main text: T is the MPC horizon, K is the number of neighbor GSFs in $\mathbf{o}^t_{i,\text{neighbor}}$, H_c is the HOCBF horizon, and $\lambda^t_i = [w^t_{p,i}, w^t_{u,i}, d^t_{\text{safe},i}]^\top$ is the adaptive parameter vector.

A.6 Compute-Cost Profile

Using the controller, learning, and evaluation settings summarized in Tables 5, 6, and 7, we profile the per-robot per-step controller time in the `random_polygon` scenario using three episodes per cell, a maximum of 50 steps per episode, and seed 100. Each sample is the wall-clock duration of one method decision call divided by the number of robots. Table 8 reports mean and standard deviation over timed steps for the robot counts used in the main evaluation.

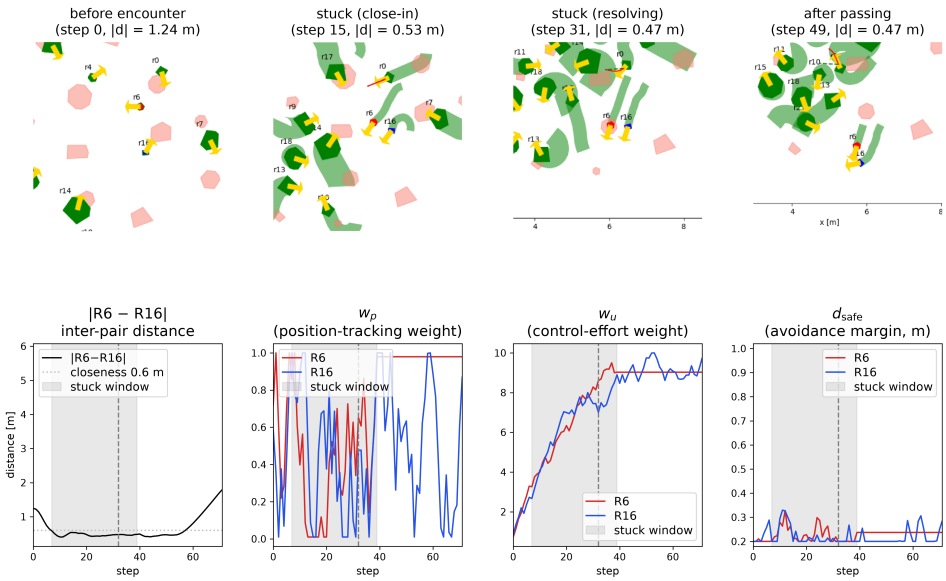


Figure 8: Reactive pair example in the random polygon scenario. The upper row shows the interaction between R6 and R16 at representative time steps, and the lower row shows the corresponding changes in inter-robot distance, w_p, w_u, and d_{safe}, indicating that the learned policy adapts the MPC parameters to the current neighbor geometry.

Table 5: Controller and scenario parameters used by `SRL-MPC`.

Group	Symbol or item	Value
Prediction model	T, Δt	5, 0.1 s
HOCBF-MPC	K, $\lvert\mathcal{N}_i^{\text{cbf}}\rvert$, H_c, distance horizon, γ, ρ_{obs}	5, 3, 3, 1, 0.9, 100
Adaptive parameters	λ_i^t	Initial: $(1.0, 1.0, 0.2\,\text{m})$; $\Lambda = [0.01, 1.0] \times [0.1, 10.0] \times [0.2, 1.0]$; $w_\theta = 0.01$ fixed
Solver and geometry	Q_1, Q_2	`Shapely`, Splitting Conic Solver (`SCS`), one outer iteration; distance cost disabled for `SRL-MPC`
Training world	N_{train}, workspace, $\mathbb{C}_i$	15 robots; 10 m × 10 m; convex polygon radius $[0.1, 0.4]$ m; irregularity $[0.1, 1.0]$; goal threshold 0.3 m

A.7 Out-of-Distribution Evaluation Details

Table 2 evaluates the same trained `SRL-MPC` policy without retraining in five settings: nonconvex union footprints; cross geometry with elongated rectangles, circles, and convex polygons; circular exchange; through traffic with five controlled robots and five nonreciprocal moving obstacles; and heterogeneous dynamics with five differential-drive polygon robots and five omnidirectional circular robots. Each method and setting uses 100 episodes, evaluation base seed 100, and at most 500 control steps. The OOD protocol derives paired scene seeds independently for each episode. The one- and two-iteration `SRL-MPC` runs use the same checkpoint and paired seeds. For nonconvex geometry, the implementation computes separation directions from the original geometry while using convex-hull-equivalent support values. Structured interactions remain challenging: `SARL` performs best in circular exchange, while `ORCA` and `AVOCADO` have higher success and fewer collisions in through traffic. Increasing the iteration count improves cross and circular exchange outcomes but reduces success in the other three settings.

A.8 Outer-Iteration Ablation and Residual Diagnostics

Table 9 compares outer-iteration counts in the random-polygon setting. Each outcome cell contains 100 episodes with base seed 100 and at most 500 steps. Within each method and robot count, only the iteration count changes and episode seeds are paired. Controller times come from separate profiles with 150 sampled control steps per cell, except the 25-robot, two-iteration cell, which has 141 usable records; each step's controller time is normalized by the robot count. Increasing iterations can reduce observed collisions while increasing computation and timeouts. With 20 robots and four iterations,

Table 6: Reinforcement learning and neural-network parameters.

Group	Symbol or item	Value
Observation	$\mathbf{o}_i^t$, $\mathbf{o}_{i,\mathrm{neighbor}}^t$	$79 = 4 + 3 \times 5 \times 5$
Action	$\Delta\lambda_i^t$	$[-0.5, 0.5]$, $[-0.5, 0.5]$, $[-0.1, 0.1]$ m; absolute values clipped to Λ
Network	CNN, actor, critic	Convolution $3 \to 16 \to 32$; actor $36 \to 64 \to 6$; critic $36 \to 64 \to 1$; log-scale clamp $[-5, 0]$
PPO schedule	Frames, batch, mini-batch, epochs	10^7, 1000, 500, 3
PPO and generalized advantage estimation (GAE) parameters	γ_{PPO}, λ_{GAE}	0.99, 0.95; clip 0.2; entropy 0.01; learning rate 3×10^{-4}; max grad norm 1.0
Training reward	r_{arr}, r_{col}, r_{safe}, r_{step}	40, 80, 1, 0.05
Safety reward	$d_{\mathrm{safe}}^{\min}$, $d_{\mathrm{safe}}^{\max}$, ϵ_d	0.2 m, 1.0 m, 0.01 m

Table 7: Evaluation and baseline settings.

Group	Symbol or item	Value
Evaluation sweep	N, n_{ep}, $E_{\max}$	$N \in \{10, 15, 20, 25\}$; $n_{\mathrm{ep}} = 100$; $E_{\max} = 500$ steps (50 s)
Randomization	Train/test seeds	Training seed 0; evaluation seed 100; disjoint scenario sequences
Scenario	`random_polygon`	Random start-goal pairs; random convex polygon footprints; collision mode `stop`
Ablation methods	`dist_mpc`, `manual_mpc`, `rule_mpc`	Fixed $w_p = 0.01$, $w_\theta = 0.01$, $w_u = 10.0$; `rule_mpc` uses $d_{\mathrm{safe}} = 0.30$ by default and switches to $(w_p, w_u, d_{\mathrm{safe}}) = (1.0, 0.1, 0.20)$ after 10 stuck steps
VO baselines	`ORCA`, `AVOCADO`, `VO-polytope`	Neighbor distance 15 m; time horizon 0.5 s; safety buffer 0.1 m; `VO-polytope` neighbor distance 3 m
RL baselines	`RL-RVO`, `SARL`	Pretrained policies; desired speed 1.0 m/s; $\Delta t = 0.1$ s

`manual_mpc` and `SRL-MPC` both have zero observed collisions, but their timeout rates are 71% and 0%, respectively. `SRL-MPC` success decreases again at five iterations, so navigation performance is not monotonic in iteration count.

A.9 Variation Across PPO Training Seeds

Table 10 reports the additional evaluation across three independently trained policies, using PPO training seeds 1, 2, and 3. Values are means and sample standard deviations across policies, rather than across individual episodes.

A.10 Quantitative Hardware Evaluation

Six robots navigate in a shared workspace over 20 paired trials per method, for 40 trials in total. Corresponding trials use the same start positions and goals sampled with the same seed. Navigation time and average speed are computed over successful trials. Table 11 reports success, collision, navigation, and controller-computation metrics. The 25-percentage-point observed success improvement is descriptive; a two-sided exact McNemar test yields $p = 0.125$. The physical robots have the same chassis and footprint; heterogeneous footprint sizes are configured in their controllers. This tests deployment with different configured geometry parameters, while validation on physically heterogeneous robots remains future work.

A.11 Computation Analysis

We compare the cost of computing the same GSF using GEOS ShortestLine and a dual support-function program implemented with CvxpyLayer/SCS. Table 12 uses batches of 120 queries, corresponding to 24 neighbors over five horizon steps before neighbor selection in the 25-robot setting. Timings exclude geometry generation, optimization-layer construction, and warm-up. Both backends pass the benchmark's numerical agreement checks on separated convex polygons. This isolates GSF computation rather than measuring an entire MPC controller. Table 13 gives the mean module breakdown for one `SRL-MPC` iteration with 20 robots. The separately measured controller total can differ slightly from the sum of rounded module values.

Table 8: Per-robot per-step controller compute time in milliseconds.

Method	$N = 10$	$N = 15$	$N = 20$	$N = 25$
`manual_mpc`	5.482 ± 1.585	6.586 ± 1.715	8.488 ± 1.418	9.955 ± 1.803
`dist_mpc`	5.202 ± 1.449	7.313 ± 1.125	8.693 ± 1.688	11.260 ± 1.358
`ORCA`	0.021 ± 0.003	0.022 ± 0.001	0.024 ± 0.002	0.026 ± 0.002
`AVOCADO`	0.022 ± 0.002	0.026 ± 0.002	0.028 ± 0.002	0.028 ± 0.001
`SARL`	15.612 ± 0.904	16.353 ± 1.810	17.346 ± 1.670	18.561 ± 1.737
`RL-RVO`	0.357 ± 0.048	0.366 ± 0.037	0.370 ± 0.033	0.377 ± 0.041
`SRL-MPC` (ours)	5.786 ± 2.023	6.729 ± 1.669	9.365 ± 1.679	10.340 ± 1.801

Table 9: Outer-iteration ablation. Outcome rates are percentages; time is mean controller latency in ms per robot and control step.

Robots	Method	L	Success	Collision	Timeout	Time
20	`manual_mpc`	1	81	6	13	8.723
20	`manual_mpc`	2	46	2	52	17.551
20	`manual_mpc`	3	41	1	58	25.944
20	`manual_mpc`	4	29	0	71	34.011
20	`manual_mpc`	5	41	0	59	41.670
20	`SRL-MPC`	1	94	4	2	9.500
20	`SRL-MPC`	2	99	1	0	18.464
20	`SRL-MPC`	3	96	0	4	26.884
20	`SRL-MPC`	4	100	0	0	36.703
20	`SRL-MPC`	5	94	4	2	45.706
25	`SRL-MPC`	1	89	8	3	11.728
25	`SRL-MPC`	2	89	4	7	24.661
25	`SRL-MPC`	3	94	3	3	34.360
25	`SRL-MPC`	4	91	2	7	41.797
25	`SRL-MPC`	5	94	0	6	57.573

Table 10: Performance across three PPO training seeds. Entries are mean $\pm$ sample standard deviation in percentage points.

Setting	Success (%)	Collision (%)	Timeout (%)
Random polygon, $N = 20$	91.0 ± 2.6	7.3 ± 2.5	1.7 ± 1.2
Random polygon, $N = 25$	89.7 ± 0.6	8.3 ± 2.1	2.0 ± 1.7
Nonconvex union	97.7 ± 0.6	2.3 ± 0.6	0.0 ± 0.0
Cross geometry	77.3 ± 1.8	20.7 ± 1.6	2.0 ± 1.0
Circular exchange	56.0 ± 0.0	44.0 ± 0.0	0.0 ± 0.0
Through traffic	90.7 ± 1.5	9.0 ± 1.7	0.3 ± 0.6
Heterogeneous dynamics	95.0 ± 1.7	0.3 ± 0.6	4.7 ± 2.1

Table 11: Hardware results with 20 trials per method. Controller time is in ms per robot and step.

Method	Success (%)	Collision (%)	Navigation time (s)	Speed (m/s)	Controller time (ms)
`ORCA`	60	40	21 ± 3.14	0.43 ± 0.09	0.078
`SRL-MPC`	85	15	17 ± 2.15	0.48 ± 0.12	8.53

Table 12: GSF backend latency for 120 queries. Times are medians in ms per batch. The last column is the median of paired optimization/GEOS timing ratios.

Edges	GEOS	CvxpyLayer/SCS	Median paired ratio
3	0.102	11.139	108.37×
4	0.111	13.066	119.34×
8	0.142	35.225	263.94×
16	0.196	127.758	641.92×
32	0.343	550.835	1560.72×

Table 13: One-iteration `SRL-MPC` module profile with 20 robots: mean ms per robot and control step over 150 sampled steps.

Module	Time
RL policy and parameter application	0.058
Path update	0.206
Nominal trajectory update	0.019
Q_1: GSF computation	5.994
Q_2: MPC optimization	3.188
Other operations	0.034
Total	9.500

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contribution as `SRL-MPC`, a shape-aware reinforcement learned MPC framework for dense heterogeneous navigation. The claims are limited to the proposed GSF-based HOCBF formulation, the decomposed local MPC problem, RL-based online parameter adaptation, randomized simulated experiments with convex polygonal objects, robustness tests, out-of-distribution evaluations, variation across training seeds, and quantitative hardware evaluation.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated limitations subsection in Appendix A.2. It discusses the computational cost of solving the local HOCBF-MPC subproblem, the scope of the randomized convex-polygon evaluation, and assumptions on reliable state estimates, short-horizon nominal predictions, and convex or convex-decomposed footprints.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present a formal theorem or proof. The mathematical development is formulation-based, and the manuscript states the modeling assumptions, including convex body-frame occupied sets, affine kinematic dynamics, fixed geometric features in the decomposed local subproblem, and a soft residual form for the deployed local HOCBF-MPC subproblem.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The main text and Appendix A.5 provide the scenario generation, robot counts, evaluation protocol, controller parameters, training settings, baseline settings, and compute profile needed to reproduce the reported results to the extent possible without released code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The current submission does not provide public code or model weights. The experiments use procedurally generated randomized simulation scenarios rather than a fixed external dataset, and the paper reports the main generation and evaluation settings for reproducibility. The authors plan to open the code in the future.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix A.5 reports the controller, scenario, reinforcement learning, neural-network, evaluation, and baseline settings. The main experiments section also specifies the train-test seed separation, robot counts, number of episodes, episode length, metrics, hardware, and the PPO-based training procedure.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The baseline and ablation tables report episode-level success, collision, and timeout percentages over 100 episodes per robot count. Continuous metrics such as navigation time, speed, path length, and compute time are reported as mean values with standard deviations, as stated in the experiments section; additional results report sample standard deviations across three PPO training seeds (Table 10).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The experiments section reports the compute platform used for the main experiments and the approximate policy training time. Appendix A.6 reports the measured per-robot controller time across robot counts.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is a robotics navigation study evaluated primarily in simulation with quantitative hardware evaluation, and it does not involve human-subject experiments, private data, scraped datasets, or high-risk released models. The experimental claims are scoped to the reported evaluation.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix A.3 discusses potential positive impacts for safer robot navigation and potential negative impacts from premature deployment, state-estimation errors, model mismatch, and computation delays.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk pretrained model, scraped dataset, or generative system. The proposed method is a robot-navigation controller evaluated in simulation and on physical robots, so no special release safeguards are required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the prior methods, baselines, solver/library components, and geometric formulations that the method and experiments build on. The reported scenarios are procedurally generated, and the submission does not redistribute third-party code, data, or model assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release a new dataset, benchmark, or model asset as part of the submission. The randomized scenarios and trained policy are used for evaluation but are not submitted as standalone assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The experiments are conducted in simulation and with six physical robots. No crowdsourcing, human-subject data collection, or participant interaction is involved.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The experiments do not involve human subjects, participant data, or human-subject interventions. Therefore, IRB approval or an equivalent human-subjects review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The `SRL-MPC` framework does not use LLMs as part of the method, experiments, or scientific contribution. Any language-editing assistance does not affect the core methodology or results.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.