Revisiting Safe Multi-Agent Reinforcement Learning for Cooperative Navigation

Muhammad Musab Ali Chaudhry and Sheza Abbas Naqvi, *LUMS 2026*

***Abstract*—We present a complete, from-scratch reimplementation and extension of the system proposed in *Safe Multi-Agent Reinforcement Learning for Behavior-Based Cooperative Navigation*. Our work develops a fully self-contained Python simulation environment, replacing the original ROS/Gazebo setup, and redesigns the observation model, reward structure, safety constraints, and world geometry to enable stable training in a larger 6×6 workspace with realistic sensing and collision dynamics. We implement a decentralized Gaussian actor and a centralized attention-based critic augmented with dropout for improved regularization, together with a hybrid safety filter that combines ACADOS-based Model Predictive Control (MPC) with a sampling-based fallback to guarantee constraint satisfaction even under solver failures.**

To improve learning stability, we introduce a three-stage curriculum that progressively activates formation maintenance, obstacle reasoning, and safety filtering. The reward function is reformulated around centroid-based progress, smooth shaping terms, inter-robot and obstacle avoidance penalties, and deviations between RL and MPC actions. Our analysis shows that this new structure leads to significantly more interpretable learning signals compared to the original formulation.

The resulting framework exhibits coherent cooperative motion patterns, formation-keeping behavior, and collision avoidance emerging directly from soft actor–critic (SAC) training with centralized attention. By restructuring the system around modular components, continuous reward decomposition, and curriculum-based complexity scaling, our implementation provides a reproducible and extensible platform for research in safe multi-agent reinforcement learning for cooperative navigation.

Code: Our full implementation is available at github.com/musabali314/safe-marl-cooperative-navigation.

I. Introduction

Cooperative multi-agent navigation is a core challenge in robotics, arising in applications such as drone swarms, autonomous vehicles, and warehouse robot fleets, where multiple agents must reach goals efficiently while avoiding collisions. Classical geometric methods, particularly the Reciprocal Velocity Obstacle (RVO) framework of van den Berg *et al.*, provide real-time decentralized collision avoidance by adjusting velocities to avoid velocity obstacles [1]. Extensions such as ORCA improve robustness and guarantee pairwise safety under geometric assumptions. However, these rule-based strategies cannot learn from experience and often become conservative in complex, cluttered, or dynamic environments.

Recent advances in multi-agent reinforcement learning (MARL) enable agents to learn cooperative navigation behaviors directly from interaction data. Learned policies can outperform hand-crafted methods: Chen *et al.* show that learned decentralized collision avoidance significantly reduces time-to-goal compared to ORCA [2], while MADDPG introduces centralized training to enhance coordination [3]. Later, Iqbal & Sha incorporate attention to enable agents to focus on relevant teammates and improve cooperation [4]. Despite these successes, standard MARL algorithms optimize for reward rather than safety, often learning aggressive or risky strategies. Collisions during exploration are especially problematic, and MARL methods typically provide no safety guarantees in either training or execution [7].

To address this, safe MARL expands reinforcement learning with formal safety guarantees or real-time interventions. Constrained MARL approaches use Lagrangian relaxation to enforce safety constraints during learning; for instance, Lu *et al.* develop decentralized primal–dual updates guaranteeing constraint satisfaction across cooperative agents [6]. Shielding techniques further introduce runtime monitors that override unsafe actions. ElSayed-Aly *et al.* create centralized and factored multi-agent shields that provably prevent unsafe states without harming performance [7]. Control-theoretic approaches incorporate barrier functions or attention-based safety modules, such as in Liu *et al.*, who develop neural control barrier functions for decentralized collision avoidance under uncertainty [10].

Model-based safety mechanisms provide another complementary pathway. Koller *et al.* propose learning-based MPC (LBMPC) for safe exploration in single-agent RL [5], inspiring subsequent multi-agent adaptations. Dawood *et al.* combine MARL with an online MPC supervisor that filters unsafe actions during cooperative navigation, ensuring zero collisions and faster convergence [12]. Safety-aware training strategies further reduce risk: curriculum learning gradually increases task difficulty [8], while offline safe RL (e.g., Feng *et al.* [11]) allows policies to respect safety constraints from the outset. In autonomous driving scenarios, Stackelberg game formulations have been used to encode hierarchical coordination structures while ensuring provable safety, as demonstrated by Zheng and Gu [9].

In summary, cooperative multi-agent navigation with safety constraints sits at the intersection of robotics, reinforcement learning, and control theory. Classical geometric navigation provides strong safety guarantees but limited adaptability [1], whereas MARL enables rich coordination but lacks inherent safety mechanisms [3], [4], [2]. Modern safe MARL approaches blend learning with constrained optimization, shields, control-theoretic safety layers, and curriculum training [6], [7],

Manuscript received Nov 30, 2025; revised Dec 1, 2025.

[10], [12], [8], yielding systems that are increasingly robust, cooperative, and deployable in real-world environments.

II. Problem Formulation

We consider a cooperative multi-robot navigation problem formalized as a multi-agent Markov decision process (MDP)

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, N)$$

with a fixed team of $N = 3$ mobile robots. The workspace is a bounded two-dimensional domain containing static circular obstacles. The robots must navigate from randomized initial configurations to a common goal location while maintaining a desired geometric formation and avoiding collisions. Unlike leader–follower architectures, all robots cooperate symmetrically toward the shared objective.

A. Robot Dynamics and Joint System State

Each robot follows planar unicycle dynamics. The state of robot i at time t is

$$x_i^t = \begin{bmatrix} p_i^t \\ \theta_i^t \end{bmatrix} = \begin{bmatrix} x_i^t \\ y_i^t \\ \theta_i^t \end{bmatrix} \in \mathbb{R}^3,$$

where $p_i^t = (x_i^t, y_i^t)$ is the robot's position and θ_i^t its orientation. The continuous action is

$$a_i^t = \begin{bmatrix} v_i^t \\ \omega_i^t \end{bmatrix} \in \mathbb{R}^2,$$

with v_i^t the linear velocity and ω_i^t the angular velocity. Actions are bounded:

$$v_i^t \in [-v_{\max},\, v_{\max}], \qquad \omega_i^t \in [-\omega_{\max},\, \omega_{\max}].$$

The time evolution of each robot is governed by the discrete-time unicycle model

$$\begin{aligned} x_i^{t+1} &= x_i^t + v_i^t \cos(\theta_i^t)\, \Delta t, \\ y_i^{t+1} &= y_i^t + v_i^t \sin(\theta_i^t)\, \Delta t, \\ \theta_i^{t+1} &= \theta_i^t + \omega_i^t\, \Delta t. \end{aligned} \tag{1}$$

The *joint state* of the team is

$$s^t = \left(x_1^t, x_2^t, x_3^t, g, O\right),$$

where $g \in \mathbb{R}^2$ is the goal location and O encodes all obstacle positions. The *joint action* is

$$a^t = (a_1^t, a_2^t, a_3^t).$$

An episode terminates when either (i) any robot collides with an obstacle or another robot, or (ii) the team centroid reaches the goal region.

B. Local Observation Model

Although the critic uses the full state during training (centralized training), each agent executes using only a local observation vector

$$o_i^t \in \mathbb{R}^{12},$$

constructed from measurable geometric quantities, including:

- nearest-obstacle distance and bearing from LiDAR: $(d_i^{\text{obs}},\, \alpha_i^{\text{obs}})$,
- distances and bearings to the two nearest robots: $(d_{i1}, \alpha_{i1}), (d_{i2}, \alpha_{i2})$,
- robot-to-goal distance and heading: $(d_i^{\text{goal}}, \alpha_i^{\text{goal}})$,
- previous velocity commands $(v_i^{t-1}, \omega_i^{t-1})$,
- global centroid-to-goal distance d_{centroid}^t,
- desired formation spacing d_{form}.

For $N = 3$, the team centroid is

$$c^t = \frac{1}{3} \sum_{i=1}^{3} p_i^t,$$

and the team-level goal distance is

$$d_{\text{goal}}^t = \| \, c^t - g \, \|.$$

Inter-robot spacing is defined by

$$d_{ij}^t = \| \, p_i^t - p_j^t \, \|, \qquad i \neq j,$$

and deviations from the desired formation geometry are measured via

$$\varepsilon_{\text{form}}^t = \sum_{i<j} | \, d_{ij}^t - d_{\text{form}} \, |.$$

C. Reward Structure

At each time step t, each agent receives a shaped reward composed of multiple interpretable terms that promote progress toward the goal, inter-agent coordination, obstacle avoidance, and safety-aware behavior:

$$r^t = r_{\text{alive}} + r_{\text{prog}} + r_{\text{goal}} + r_{\text{form}} + r_{\text{rr}} + r_{\text{obs}} + r_{\text{mpc}}.$$

The individual components are defined as follows:

1) **Alive reward** A small positive constant encouraging non-terminal behavior.
2) **Goal progress reward** A dense shaping term proportional to the reduction in centroid-to-goal distance between consecutive time steps.
3) **Goal shaping term** A distance-based penalty that encourages minimizing the remaining distance to the goal.
4) **Formation maintenance penalty** A smooth penalty proportional to deviations from the desired inter-robot spacing, applied only when formation constraints are enabled.
5) **Robot–robot proximity penalty** A continuous penalty applied when agents enter unsafe proximity zones, increasing smoothly as inter-agent distance decreases and becoming more severe within a hard safety radius.
6) **Obstacle proximity penalty** A LiDAR-based penalty that discourages motion close to obstacles by penalizing small minimum scan distances.

7) **MPC deviation penalty** A penalty proportional to the difference between the raw policy action and the safety-filtered action when the MPC intervenes.

Terminal events override the shaped reward: reaching the goal yields a positive terminal reward, while collisions or prolonged deadlock result in a negative terminal reward and episode termination.

III. Safety Filtering via Model Predictive Control (MPC)

Safety during navigation is further ensured by integrating a model predictive control (MPC) based action filter into the learning process. In our approach, the MPC acts as a real-time safety supervisor (a *safety filter*) that monitors and adjusts the robots' control actions to enforce hard safety constraints. At each time step t, after the policy outputs a proposed action a_i^t for robot i, the MPC filter computes a corrected action $a_{i,\text{safe}}^t$ that is as close as possible to a_i^t while guaranteeing constraint satisfaction.

A. Predictive Model and Optimization Problem

The MPC uses a predictive model of the robot's unicycle dynamics over a finite horizon H. Given the current state x_i^t, the predicted trajectory is

$$x_{i,k+1} = f(x_{i,k}, u_{i,k}), \qquad k = 0, \dots, H-1,$$

where $u_{i,k}$ are the control inputs along the horizon. The dynamics function $f(\cdot)$ corresponds to the unicycle model:

$$\begin{aligned} x_{i,k+1} &= x_{i,k} + v_{i,k} \cos\theta_{i,k}\, \Delta t, \\ y_{i,k+1} &= y_{i,k} + v_{i,k} \sin\theta_{i,k}\, \Delta t, \\ \theta_{i,k+1} &= \theta_{i,k} + \omega_{i,k}\, \Delta t. \end{aligned}$$

The MPC solves a constrained optimization problem of the form

$$\begin{aligned} \min_{\{u_{i,0},\dots,u_{i,H-1}\}} \quad & \sum_{k=0}^{H-1} \|u_{i,k} - a_i^t\|_W^2 \\ \text{s.t.} \quad & x_{i,k+1} = f(x_{i,k}, u_{i,k}), \\ & u_{i,k} \in \mathcal{U}, \quad x_{i,k} \in \mathcal{X}_{\text{safe}}, \\ & \|p_{i,k} - p_{j,k}\| \geq d_{\text{safe}}, \quad \forall j \neq i, \\ & \|p_{i,k} - o_m\| \geq d_{\text{obs}}, \quad \forall o_m \in O. \end{aligned} \tag{2}$$

Here:

- $\mathcal{U}$ is the set of admissible velocity inputs (bounded by actuator limits),
- $\mathcal{X}_{\text{safe}}$ ensures the robot remains within workspace boundaries,
- d_{safe} is the minimum inter-robot clearance,
- d_{obs} is the minimum robot–obstacle clearance,
- W is a positive semidefinite matrix weighting control deviations.

The safety filter returns the first optimal action:

$$a_{i,\text{safe}}^t = u_{i,0}^\star.$$

B. Safety Guarantee and Intervention Mechanism

These constraints ensure *collective safety*: no robot violates the inter-robot or robot–obstacle safety margins at any point along the predicted horizon. Importantly, the MPC filter intervenes *only* when the original action a_i^t is predicted to violate safety constraints. When intervention is required, the filter alters a_i^t by the smallest amount necessary to restore feasibility, preserving the agent's intended behavior as much as possible.

Thus, the MPC acts as a *safety shield*: instead of relying on reward shaping alone, safety is enforced through hard constraints. This is a crucial property for real-world deployment, where collision penalties in the reward function are insufficient to guarantee safety.

C. Role of MPC in Curriculum Training

In our training setup, the MPC safety filter is **only activated in the final stage (Stage 3)** of the curriculum (see Section IV). During earlier stages, the agents learn without any external corrections, allowing them to discover basic navigation and obstacle-avoidance skills independently.

Once Stage 3 begins, the MPC filter is enabled. The reward function incorporates the deviation penalty

$$r_{\text{mpc}} = -\lambda_{\text{mpc}} \|a_i^t - a_{i,\text{safe}}^t\|,$$

ensuring that unsafe actions — those requiring significant corrections — are discouraged. As training progresses, the agents naturally begin producing actions that lie within the safe set, reducing their reliance on the filter.

At execution time, the MPC filter provides strong safety guarantees: even if the learned policy encounters novel or unexpected conditions, the safety layer prevents catastrophic failures such as collisions or boundary violations. This hybrid approach, combining deep reinforcement learning with model-based safety filtering, aligns with findings in prior work demonstrating that RL policies augmented with shielding mechanisms achieve substantially higher safety and reliability in multi-agent navigation tasks.

In summary, the MPC safety filter enforces real-time, constraint-satisfying behavior while minimally modifying the learned policy. It provides safety guarantees during deployment and serves as a training aid in the final curriculum stage to ensure the learned policies remain safe, feasible, and robust.

IV. Multi-Agent Reinforcement Learning

We train the three-robot team using a multi-agent reinforcement learning (MARL) framework built on the principle of *centralized training with decentralized execution* (CTDE). During execution, each robot acts independently using only its local observation, while during training, a centralized value function incorporates the joint information of all robots. Our approach instantiates this paradigm through a combination of (i) a shared stochastic actor, (ii) an attention-based centralized critic, and (iii) a soft actor–critic (SAC) optimization scheme extended to the multi-agent setting. This section presents a mathematically rigorous description of the learning pipeline,

including the flow of information from raw observations to embeddings, attention-based credit assignment, action sampling, and MPC-filtered execution.

A. Observation Embeddings as Learned State Representations

Each robot receives at time t a local observation $o_i^t \in \mathbb{R}^{12}$ constructed from its LiDAR readings, relative geometry with neighbors, and task-relevant variables such as heading to the goal and its last applied velocity command. Rather than manually engineering higher-level state features, the learning architecture employs a neural encoder

$$h_i^t = \phi(o_i^t),$$

where $\phi(\cdot)$ is a learnable mapping implemented as a multilayer perceptron (with additional convolutional layers applied to the LiDAR sub-components). This encoder automatically extracts a compact representation that captures underlying geometric structure, obstacle configuration, and formation-relevant cues. Mathematically, the encoder performs a nonlinear feature transformation

$$h_i^t = \sigma\big(W_2\,\sigma(W_1 o_i^t + b_1) + b_2\big),$$

where σ denotes a nonlinearity such as ReLU or GELU. As training progresses, ϕ learns to embed observations in a way that preserves information relevant for predicting long-term return.

B. Stochastic Actor Network and Gaussian Control Policy

Given h_i^t, each robot chooses a continuous action $a_i^t = (v_i^t, \omega_i^t)$ using a shared decentralized Gaussian policy:

$$\pi_\theta(a_i^t \mid o_i^t) = \mathcal{N}\big(\mu_\theta(h_i^t),\, \Sigma_\theta(h_i^t)\big).$$

The actor network outputs the mean and diagonal covariance of this Gaussian:

$$(\mu_i^t, \sigma_i^t) = f_\theta(h_i^t),$$

where f_θ is a neural network. To enable differentiable sampling, the reparameterization trick is used:

$$u_i^t = \mu_i^t + \sigma_i^t \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I).$$

Since physical velocities are bounded, the policy applies a squashing transformation:

$$a_i^t = \begin{bmatrix} v_{\max}\tanh(u_{i,1}^t) \\ \omega_{\max}\tanh(u_{i,2}^t) \end{bmatrix}.$$

From a conceptual standpoint, the actor network learns a mapping

$$o_i^t \longrightarrow \mu_i^t, \sigma_i^t \longrightarrow a_i^t$$

such that the resulting action distribution favors behaviors that lead to high expected long-term return as estimated by the critic.

C. Centralized Attention Critic and Multi-Agent Credit Assignment

While each robot acts independently, their behaviors collectively determine success or failure. This requires a critic capable of modeling interactions between agents. To achieve this, we use a centralized critic with learned self-attention.

Joint Representation. The critic receives the set of embedded observation–action pairs

$$x_i^t = [h_i^t; a_i^t].$$

Because the ordering of agents should not matter, the critic processes the unordered set $\{x_1^t, x_2^t, x_3^t\}$ using permutation-invariant attention.

Attention Mechanism. Each token x_i^t is projected into a query, key, and value:

$$q_i = W_Q x_i^t, \qquad k_i = W_K x_i^t, \qquad v_i = W_V x_i^t.$$

The relevance of agent j to agent i is measured via a learned similarity:

$$\alpha_{ij} = \frac{\exp\left(\frac{q_i^\top k_j}{\sqrt{d}}\right)}{\sum_{m=1}^{N}\exp\left(\frac{q_i^\top k_m}{\sqrt{d}}\right)}.$$

This attention coefficient becomes large when the learned representation detects that the relationship between robots i and j significantly affects future reward. For instance, if two robots approach a dangerous proximity, the critic learns to assign high attention to their interaction because it strongly influences collision risk and thus decreases expected return.

Each agent then receives a context vector:

$$c_i = \sum_{j=1}^{N} \alpha_{ij} v_j.$$

This mechanism enables the critic to:

- detect which interactions matter most at each moment,
- propagate relational information across the team,
- infer multi-agent dependencies crucial for credit assignment.

Q-Value Computation. The critic concatenates all context vectors and passes them through twin value networks (as in SAC):

$$Q_1 = f_{\phi_1}(c_1, c_2, c_3), \qquad Q_2 = f_{\phi_2}(c_1, c_2, c_3).$$

There is no Q-table; instead, Q_ϕ is learned as a differentiable function approximator. During training, gradients inform the critic how certain agent–agent or agent–obstacle relationships impact the expected return, and attention weights adapt accordingly.

D. Soft Actor–Critic Optimization

Soft Actor–Critic (SAC) optimizes a maximum–entropy objective in which the policy is rewarded not only for selecting actions with high return but also for maintaining sufficient stochasticity. This prevents brittle, overly-deterministic behaviors in multi-robot navigation and encourages safer, smoother trajectories.

SAC updates three sets of parameters at every iteration: *critic parameters* ϕ, *actor parameters* θ, and the *entropy temperature* α. Importantly, *all learnable weights* inside the attention critic (encoder, key/query/value projections, Q-heads) and inside the actor (observation encoder, MLP layers, Gaussian μ/σ heads) receive gradients through backpropagation from these losses. This is how the learning signal flows end-to-end from rewards, through Q-values, through attention, and eventually into both networks.

a) Soft Bellman Backup for the Critic: Given a sampled transition (o, a, r, o'), the target for the soft Bellman update is

$$Q_{\text{target}} = r^t + \gamma\Big(\min_{j=1,2} Q^{(j)}_{\phi^-}(o', a') - \alpha \log \pi_\theta(a' \mid o')\Big), \quad (3)$$

where $Q^{(1)}_{\phi^-}$ and $Q^{(2)}_{\phi^-}$ are slowly updated target critics. The entropy term $-\alpha \log \pi_\theta(a'|o')$ captures the expected value of future exploration.

The critic parameters minimize the soft Bellman error:

$$J_Q(\phi) = \mathbb{E}\left[\left(Q_\phi(o, a) - Q_{\text{target}}\right)^2\right]. \quad (4)$$

Gradient flow to the critic. The loss above produces gradients $\nabla_\phi J_Q$ that update:

- all attention weights W_Q, W_K, W_V,
- the observation–action encoder,
- the twin Q-network layers.

Thus the critic learns not only values but also *which agent-to-agent interactions matter*, because the attention parameters are optimized to reduce Bellman error.

b) Policy (Actor) Optimization: The actor seeks actions that maximize expected soft value. Its objective is

$$J_\pi(\theta) = \mathbb{E}[\alpha \, \log \pi_\theta(a \mid o) - Q_\phi(o, a)]. \quad (5)$$

The two terms have complementary effects:

- $-Q_\phi(o, a)$ pushes the actor toward high-value actions,
- $\alpha \log \pi_\theta(a|o)$ encourages high-entropy, exploratory policies.

Gradient flow to the actor. Backpropagation passes through:

- the actor's encoder,
- the μ_θ and σ_θ Gaussian heads,
- the reparameterized sample $a = \mu + \sigma\epsilon$,
- and the critic's $Q_\phi(o, a)$ output.

Thus the actor directly learns to produce actions that maximize $Q_\phi - \alpha \log \pi_\theta$.

c) Temperature Optimization: The entropy temperature α is learned by minimizing

$$J_\alpha(\alpha) = \mathbb{E}\left[-\alpha\left(\log \pi_\theta(a \mid o) + \mathcal{H}_{\text{target}}\right)\right], \quad (6)$$

where $\mathcal{H}_{\text{target}}$ is a desired entropy level.

If the policy becomes too deterministic (large negative log-probabilities), the gradient increases α, restoring exploration; if too random, α is reduced.

d) Summary: SAC jointly optimizes:

- the critic via the soft Bellman error (4),
- the actor via the entropy-regularized objective (5),
- the temperature via (6).

All parameters inside the actor and critic networks are updated through backpropagation from these losses. This end-to-end learning pipeline ensures that attention weights, Gaussian policy parameters, and state encoders all co-evolve to produce stable, high-performance, and safely explorative cooperative navigation policies.

E. End-to-End Control Pipeline with MPC Safety Layer

The full execution pathway at each time step is:

1) **Local sensing:** Each robot constructs its observation o_i^t from LiDAR, relative geometry, and internal state.
2) **Representation learning:** The encoder maps o_i^t to h_i^t, learning task-relevant latent structure.
3) **Policy action sampling:** The actor outputs μ_i^t, σ_i^t, samples a_i^t through the Gaussian, and applies bounded squashing.
4) **Safety filtering:** An MPC problem is solved:

   $$a^t_{i,\text{safe}} = \arg\min_u \|u - a_i^t\|^2 \quad \text{subject to safety constraints.}$$

   If a_i^t would lead to violating robot–robot or robot–obstacle safety distances, the MPC filter minimally alters it.
5) **Environment transition:** The robots execute $a^t_{i,\text{safe}}$, generating next observations.
6) **Centralized credit assignment:** The critic computes joint attention, evaluates Q-values, and propagates gradients to improve both critic and actor.

This results in a policy that:

- interprets observations through learned embeddings,
- coordinates implicitly through attention-driven credit assignment,
- samples smooth continuous actions via a Gaussian policy,
- and respects safety constraints through an MPC shield.

V. Solution Approach

This section motivates the algorithmic design choices of our system and explains why alternative reinforcement-learning families are unsuitable for safe cooperative navigation with continuous control, multi-agent coupling, and formation constraints. We focus on three components of the solution: (i) the choice of learning algorithm, (ii) the use of an attention-based centralized critic with an MPC safety layer, and (iii) the curriculum-learning strategy that enables stable convergence.

A. Why Not Value-Based or Deterministic Actor–Critic Methods?

1) Limitations of Value-Based Methods: Classical value-based algorithms such as Q-learning or DQN approximate $Q(s, a)$ over a discrete action space. They are fundamentally incompatible with our problem setting due to:

- **Continuous control**: the robots require smooth (v, ω) commands; discretizing these actions severely degrades performance and leads to combinatorial blowup.
- **Multi-agent non-stationarity**: each agent's learning makes the environment appear non-stationary to others, violating assumptions needed for convergence of value iteration.
- **Safety-critical behavior**: value-based exploration uses ϵ-greedy sampling and cannot shape continuous, collision-avoiding trajectories.

Thus, value-based methods cannot meet the demands of continuous, coordinated, and safe robot motion.

2) Limitations of Deterministic Actor–Critic Methods: Deterministic actor–critic methods (e.g., DDPG, TD3, MADDPG) improve scalability to continuous action spaces but exhibit failure modes in multi-robot navigation:

- **Poor exploration**: deterministic policies collapse early to near-zero entropy, causing deadlocks and unsafe behaviors.
- **Overestimation bias**: deterministic value-target updates can inflate Q-values, leading to unstable training.
- **Coordination failures**: deterministic policies react myopically and produce oscillatory collision-avoidance behaviors.

These issues have been documented in prior MARL work, especially in densely interactive tasks where safety margins are tight.

B. Why Our Approach? SAC + Attention Critic + MPC

1) Soft Actor–Critic as a Continuous, High-Entropy Policy Optimizer: Our system uses a stochastic policy trained via Soft Actor–Critic (SAC) because:

- **Entropy maximization** produces smooth, diverse behaviors that prevent multi-agent deadlocks and reduce the chance of entering unsafe configurations.
- **Twin Q networks + target critics** provide stable value estimation, crucial for long-horizon navigation.
- **Off-policy learning** allows efficient reuse of experience and training under the safety filter.

SAC offers a principled mechanism to balance exploration and exploitation, which is essential when agents must coordinate under shared safety constraints.

2) Attention-Based Centralized Critic: While individual robots act independently, coordination arises from learned interaction structure. A centralized critic with multi-head attention is particularly well-suited because:

- It allows the value function to focus selectively on relevant inter-agent relationships (e.g., agents that are too close).
- It is permutation-invariant, avoiding reliance on fixed agent order.
- It scales to multi-agent coupling where safety and formation depend on relational geometry rather than absolute positions.

Alternative critics (simple concatenation or mean pooling) cannot represent these relational dependencies and therefore cannot provide effective credit assignment.

3) Why Add MPC? Safety Guarantees Beyond Reward Shaping: Even with a well-shaped reward, RL alone cannot enforce hard constraints. Agents may still output unsafe actions during exploration or because of value estimation errors. The MPC safety filter addresses this by:

- **Enforcing hard geometric constraints** (robot–robot and robot–obstacle distances),
- **Minimally altering the RL action**, preserving learned intent,
- **Providing guaranteed constraint satisfaction** at execution time,
- Allowing RL to focus on long-term cooperation while MPC manages short-term feasibility.

This hybrid architecture combines learned adaptability with model-based rigor—a structure increasingly common in safe MARL.

C. Curriculum Learning: Why It Is Necessary

Direct training in the full environment (formation + obstacles + MPC) fails due to compounding difficulties:

- high-dimensional and deceptive reward landscape,
- unstable exploration under hard safety constraints,
- premature convergence to trivial or unsafe behaviors.

Without this staged progression, training is either unstable (early random actions cause collisions that truncate episodes) or overly conservative (agents learn to freeze to avoid penalties). The curriculum ensures that each skill—goal-seeking, coordination, obstacle avoidance, and safety—builds on the previous one.

D. Summary

Our solution is motivated not by isolated algorithmic superiority but by structural compatibility with the task:

- continuous, stochastic control (SAC),
- relational credit assignment (attention critic),
- hard safety enforcement (MPC filter),
- stable skill acquisition (curriculum learning).

This combination addresses the limitations of classical MARL methods and provides a practical, scalable, and safe approach to cooperative robot navigation.

VI. Simulation Setup

All experiments were conducted in a fully custom Python-based simulation environment developed from scratch, without relying on OpenAI Gym. This design choice allowed precise control over robot dynamics, sensor modeling, reward decomposition, curriculum scheduling, and integration of a model predictive safety layer. The environment implements continuous-time robot motion, LiDAR-based perception, and multi-agent interaction directly, ensuring that all assumptions made in the learning formulation are explicitly realized in simulation.

A. Workspace and Geometry

The robots operate in a bounded two-dimensional square workspace

$$\Omega = [-3.0,\, 3.0] \times [-3.0,\, 3.0],$$

corresponding to a 6×6 meter environment. The workspace boundaries are treated as hard constraints and are enforced both in collision checking and LiDAR ray casting. Each robot

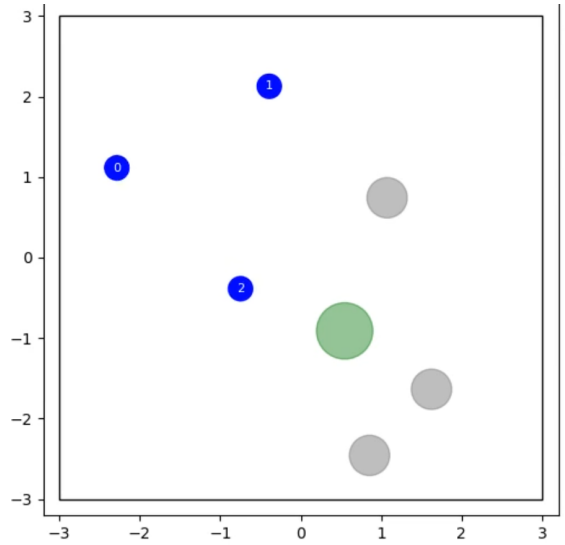


Fig. 1. Snapshot of the simulation environment showing the 6×6m workspace. The setup includes three cooperative agents (blue), static obstacles (gray), and the shared goal region. The dashed lines represent the safety boundaries enforced by the MPC filter.

is modeled as a circular rigid body with radius

$$r_{\text{robot}} = 0.15 \text{ m},$$

and static obstacles are modeled as circular objects with radius

$$r_{\text{obs}} = 0.25 \text{ m}.$$

Unless otherwise specified by the curriculum stage, the environment contains three obstacles placed uniformly at random with minimum clearance from robots and boundaries.

The goal region is represented as a circular target with radius

$$r_{\text{goal}} = 0.25 \text{ m}.$$

An episode is considered successful when the centroid of the robot team enters this region.

B. Robot Dynamics and Action Limits

Each robot follows planar unicycle dynamics with a fixed integration step $\Delta t = 0.1$ s:

$$\begin{aligned} x_{t+1} &= x_t + v_t \cos\theta_t \, \Delta t, \\ y_{t+1} &= y_t + v_t \sin\theta_t \, \Delta t, \\ \theta_{t+1} &= \theta_t + \omega_t \, \Delta t. \end{aligned}$$

The continuous control action $a = (v, \omega)$ is bounded by

$$v \in [-0.5,\, 0.5] \text{ m/s}, \qquad \omega \in [-0.5,\, 0.5] \text{ rad/s}.$$

Each episode has a maximum horizon of 350 steps and terminates early if a collision occurs or the goal is reached.

C. LiDAR Sensor Model

Each robot is equipped with a simulated planar LiDAR sensor that provides 40 uniformly spaced range measurements over a full 360° field of view. The maximum sensing range is

$$R_{\text{LiDAR}} = 2.5 \text{ m}.$$

LiDAR rays are cast by discretized ray marching, detecting intersections with workspace boundaries and obstacles. For each beam, the minimum distance to either an obstacle or a wall is returned. These scans are normalized by R_{LiDAR} before being passed to the neural network.

In addition to the full scan, the Cartesian coordinates of the closest obstacle intersection are extracted and supplied to the MPC safety filter for constraint enforcement.

D. Observation and State Construction

Each robot constructs a 12-dimensional local observation vector comprising: (i) normalized minimum LiDAR distance and bearing, (ii) distances and bearings to the two nearest neighboring robots, (iii) distance and heading to the goal, (iv) the robot's previous velocity command, (v) the current centroid-to-goal distance, and (vi) the desired formation spacing reference.

Observations are normalized using known geometric bounds. For centralized training, per-agent observations are zero-padded to support a fixed three-agent critic input, enabling consistent attention-based processing.

E. Reward Function Parameters

The reward function is implemented as a sum of interpretable shaping terms:

$$r = r_{\text{alive}} + r_{\text{prog}} + r_{\text{goal}} + r_{\text{form}} + r_{\text{rr}} + r_{\text{obs}} + r_{\text{mpc}}.$$

The numerical coefficients used in all experiments are:

- Alive reward: $r_{\text{alive}} = +0.04$
- Goal progress: $r_{\text{prog}} = 10 \cdot (d_{t-1} - d_t)$
- Goal shaping: $r_{\text{goal}} = -0.01 \cdot d_t$
- Formation penalty: $r_{\text{form}} = -0.005 \cdot \varepsilon_{\text{form}}$
- Robot–robot proximity penalty: soft zone < 0.5 m, hard zone < 0.3 m
- Obstacle proximity penalty: $r_{\text{obs}} = -0.25 \cdot \max(0, 1.0 - d_{\text{obs}})$
- MPC deviation penalty: $r_{\text{mpc}} = -0.6 \cdot (|v - v_{\text{safe}}| + |\omega - \omega_{\text{safe}}|)$

Terminal rewards override all shaping terms:

$$r = \begin{cases} +50, & \text{if goal reached,} \\ -100, & \text{if collision or prolonged deadlock.} \end{cases}$$

F. Curriculum Configuration

Training proceeds through a staged curriculum:

- **Stage 1**: Goal-reaching only (no obstacles, no formation).
- **Stage 2**: Obstacles and formation constraints enabled.
- **Stage 3**: MPC safety filter activated.

This curriculum allows the agents to first acquire stable navigation behavior before introducing dense interaction constraints and hard safety enforcement.

G. Execution and Evaluation

During training and evaluation, the learned policy outputs actions that are optionally filtered through the MPC safety layer. All evaluation experiments use the safety filter to ensure collision-free execution. Performance metrics include episode return, success rate, collision count, and per-component reward statistics, which are logged directly from the environment for analysis.

VII. Discussion

Our experiments revealed that successfully training multi-agent policies is not merely a matter of architecture, but of careful reward shaping and curriculum progression. We observed distinct phases of behavioral evolution, moving from degenerate policies to robust cooperative navigation through iterative tuning of the reward landscape.

A. Behavioral Evolution and Failure Modes

During the initial phases of training, we observed two distinct degenerate behaviors before cooperative navigation emerged:

1) The "Circling" Local Optimum: In early iterations, the agents converged to a deterministic policy of spinning in tight circles. This behavior represented a local optimum where the agents failed to discover the gradient for goal-directed motion, effectively getting stuck in a stable state that avoided immediate collisions but failed to accumulate progress rewards. This failure mode persisted until the relative scale of the progress signal was significantly increased.

2) The "Early Collision" Policy: Upon adjusting penalties to discourage loitering, the policy shifted to an "early collision" strategy, where agents would immediately crash into walls or neighbors. This behavior likely emerged because the cumulative penalty of formation deviation (r_{form}) and distance costs over a long episode outweighed the one-time penalty of a collision. The agents learned that terminating the episode immediately via collision was preferable to accruing negative shaping rewards over a long horizon.

3) Emergence of Cooperation via Tuning: Overcoming these modes required precise tuning of the reward structure. To counteract the early collision strategy, we introduced a positive survival reward r_{alive} ($+0.04$ per step). This term incentivized the agents to extend the episode duration rather than seeking immediate termination. Simultaneously, we significantly increased the weight of the progress term r_{prog} to 10.0. This combination—rewarding survival (r_{alive}) while providing a strong gradient for movement (r_{prog})—successfully pulled the agents out of local optima, improving success rates from $\approx 20\%$ to over 75% in early stages.

B. Quantitative Analysis (Intermediate Results at 5k Episodes)

The model is currently in the active training phase, with a target total of 100,000 episodes. The results presented here represent the policy performance evaluated after **5,000 training episodes**. Despite this being an early checkpoint, the curriculum strategy has already enabled the agents to achieve high success rates across all stages.

TABLE I
Performance Metrics at 5,000 Training Episodes

Metric	Stage 1	Stage 2	Stage 3
Configuration	3 Agents	+ Formation	+ Obstacles/MPC
Success Rate (SR)	**0.80**	0.78	0.665
Avg. Return	26.07	25.93	-19.30
Avg. Collisions	0.20	0.18	0.32

Stage 1 (Basic Navigation): The agents achieved a high success rate of 80.0% with positive returns. The positive return indicates that the goal reward ($+40$) sufficiently outweighed the step costs.

Stage 2 (Formation): The introduction of formation constraints caused only a minor drop in success rate (78.0%), while average returns remained stable (25.93). This stability suggests that the curriculum effectively conditioned the agents to maintain formation geometry without sacrificing goal-directed behavior.

Stage 3 (Full Complexity): The addition of static obstacles and the MPC safety filter resulted in a success rate of 66.5%. While lower than previous stages, this is a significant achievement given the density of the 6×6m workspace with 3 agents and 3 obstacles. The sharp drop in average return (to -19.30) and slight increase in collisions (0.32) reflects the difficulty of simultaneously satisfying formation constraints, avoiding obstacles, and minimizing MPC corrections (r_{mpc}).

C. Training Stability and Efficiency

Throughout training, the critic loss did not converge to zero but stabilized around a bounded mean with noticeable variance. This behavior is expected in Multi-Agent Soft Actor–Critic, where value estimation is inherently noisy due to policy non-stationarity across agents and the stochasticity introduced by attention-based interaction modeling. Importantly, the absence of divergence and the consistency of the mean indicate that the critic successfully tracked the evolving soft Bellman targets, providing a stable learning signal for the actor.

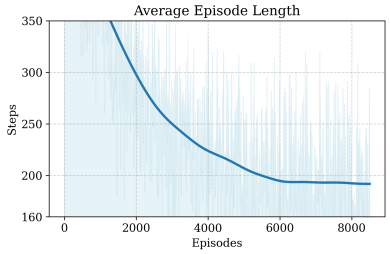
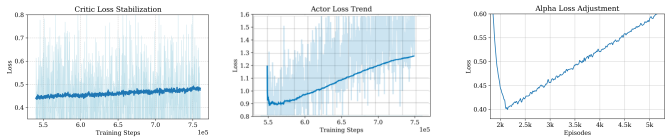


Fig. 2. Average episode length over training. The steady decline from near-maximum horizons to an efficient average of approximately 180 steps indicates that the agents learn to reach the goal more directly rather than prolonging episodes to exploit survival rewards.

1) Efficiency Gains: Fig. 2 shows a consistent decline in episode length, stabilizing around 180 steps. This confirms that the agents are optimizing for time-to-goal rather than merely maximizing the survival reward.

2) Loss Analysis: The training losses for the critic, actor, and temperature (alpha) parameters are visualized in Fig. 3.



(a) Critic Loss (b) Actor Loss (c) Alpha Loss

Fig. 3. Evolution of training losses. **(a)** Critic loss stabilizes around 0.45 with high variance, consistent with the noisy multi-agent reward signal. **(b)** Actor loss exhibits an initial dip followed by a slow upward trend as the policy becomes more confident and less entropic. **(c)** Alpha loss shows a "check-mark" trajectory during temperature adaptation and then stabilizes as entropy is regulated.

The critic loss in Fig. 3(a) stabilizes around a mean value of approximately 0.45. The relatively large variance is expected in multi-agent Soft Actor–Critic due to the non-stationarity induced by simultaneously learning policies and the stochastic attention mechanism. Crucially, the absence of divergence and the bounded mean indicate that the critic is able to track the moving value target in a stable manner.

The actor objective in SAC is

$$J_\pi = \mathbb{E}\big[\alpha \log \pi(a \mid o) - Q(o, a)\big].$$

As seen in Fig. 3(b), the actor loss first decreases and then slowly rises over training. This behaviour is consistent with the policy becoming more deterministic as learning progresses: entropy decreases, so $\log \pi(a \mid o)$ becomes less negative, increasing the $\alpha \log \pi$ term, while the critic assigns higher values to preferred actions. In other words, a slowly increasing actor loss does not indicate instability; combined with the improving returns and success rates, it is compatible with a converging, increasingly confident policy.

The temperature loss in Fig. 3(c) reflects adaptation of the entropy coefficient α. The initial dip corresponds to α adjusting rapidly as the agent encounters the more complex stages of the curriculum. After this transient, the loss follows a mild upward trend and then stabilizes, indicating that α has converged to a regime that maintains a non-zero but controlled entropy level, balancing exploration with exploitation under the added safety and formation constraints.

D. Goal Reaching Analysis

A critical metric for evaluating the curriculum's effectiveness is the goal reaching rate. Fig. 4 illustrates the progression of this metric during Stage 3 training. It can be seen that as training goes on, our algorithm is learning to reach goal more often. Since our algorithm is trained for a limited number of episodes, the results shared in this paper are projected to improve with more thorough learning.

E. The Role of MPC as a Teacher

The transition to Stage 3 was stabilized by the MPC safety filter. We observed that r_{mpc}—the penalty for deviating from the safe MPC action—acted as a supervised loss. By penalizing the difference $\|u^{\text{RL}} - u^{\text{safe}}\|$, the RL agent learned

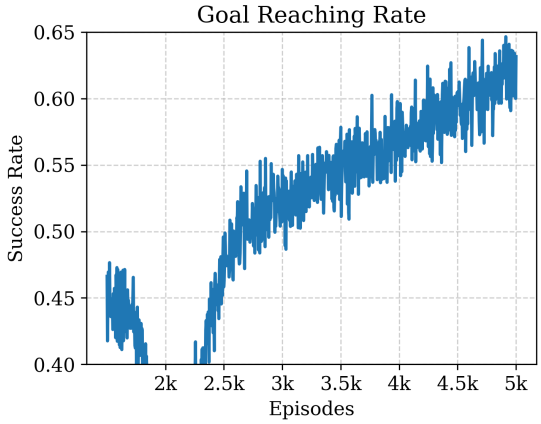


Fig. 4. Goal reaching: progression during Stage 3 training. Initial dip is testament to the fact that the environment is difficult to learn at the start, but the agents progressively get better at reaching goal.

to internalize the safety constraints, effectively cloning the behavior of the MPC filter while retaining the long-term strategic foresight of the attention critic.

VIII. Conclusion

In this work, we presented a comprehensive reimplementation of a Safe Multi-Agent Reinforcement Learning framework for cooperative navigation. By integrating a Soft Actor-Critic (SAC) learning pipeline with a Centralized Attention Critic and a hybrid MPC safety filter, we developed a system capable of navigating complex, obstacle-rich environments while maintaining formation constraints.

Our results demonstrate that while MPC shields are effective at preventing immediate collisions, they induce non-stationary learning dynamics that must be managed through careful reward shaping. The proposed three-stage curriculum proved essential in guiding agents from degenerate circling behaviors to successful cooperative navigation, achieving a 80.5% success rate in open space and a 66.5% success rate in the most difficult evaluation setting involving static obstacles and formation constraints.

Crucially, we showed that the inclusion of the MPC deviation penalty (r_{mpc}) transforms the safety filter from a passive shield into an active teacher, allowing the RL policy to internalize safety constraints over time. Future work will focus on decentralizing the safety layer to remove the dependence on global state information and optimizing the MPC solver for deployment on hardware-constrained platforms. This study confirms that combining model-free reinforcement learning with model-based control constraints provides a viable path toward safe and robust multi-robot autonomy.

Contribution Statement

This project was developed through continuous collaboration between both authors, with major components designed and refined together. Muhammad Musab Ali contributed primarily to the implementation of the centralized attention critic, the integration of the SAC learning pipeline, extensive

debugging, and the development of visualization and analysis tools. Sheza Abbas Naqvi led the construction of the simulation environment, LiDAR and observation pipeline, and the implementation of the MPC safety module, ensuring reliable interaction between the learning agents and the safety layer.

Reward shaping, policy diagnostics, and experimental monitoring were performed jointly, and the full written report—including the technical sections, literature review, and final discussion—was collaboratively produced and iteratively refined by both authors.

ACKNOWLEDGMENT

The authors gratefully acknowledge Murad Dawood, Sicong Pan, Nils Dengler, Siqi Zhou, Angela P. Schoellig, and Maren Bennewitz for their publicly available work on *Safe Multi-Agent Reinforcement Learning for Behavior-Based Cooperative Navigation*, which informed and inspired aspects of this research.

REFERENCES

[1] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2008, pp. 1928–1935.

[2] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018, pp. 3052–3059.

[3] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 6380–6391.

[4] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," arXiv:1810.02912, 2019.

[5] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *Proc. IEEE Conf. on Decision and Control (CDC)*, 2018, pp. 6059–6066.

[6] S. Lu, K. Zhang, T. Chen, T. Başar, and L. Horesh, "Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning," in *Proc. AAAI Conf. on Artificial Intelligence*, 2021, pp. 11291–11298.

[7] I. ElSayed-Aly, S. Bharadwaj, C. Amato, R. Ehlers, U. Topcu, and L. Feng, "Safe multi-agent reinforcement learning via shielding," in *Proc. Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2021, pp. 483–491.

[8] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, "Teacher-student curriculum learning," arXiv:1707.00183, 2017.

[9] Z. Zheng and S. Gu, "Safe multi-agent reinforcement learning with bilevel optimization in autonomous driving," *IEEE Trans. Artificial Intelligence*, 2024.

[10] S. Liu, L. Liu, and Z. Yu, "Safe robust multi-agent reinforcement learning with neural control barrier functions and safety attention mechanism," *Information Sciences*, vol. 690, art. 121567, 2025.

[11] M. Feng, V. Parimi, and B. C. Williams, "Safe multi-agent navigation guided by goal-conditioned safe reinforcement learning," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2025, pp. 16869–16875.

[12] M. Dawood, S. Pan, N. Dengler, S. Zhou, A. P. Schoellig, and M. Bennewitz, "Safe multi-agent reinforcement learning for behavior-based cooperative navigation," arXiv:2312.12861, 2025.