MACH3: Multi-Agent Cooperative Hierarchical Reinforcement Learning via a Three-Layer Architecture

Alessandro Bonetti[1], Xin Zhou[2], Yifeng Zhang[2*], Silvia Proia[1], Tanishq Duhan[2], Tingguang Zhou[2], Lorenzo Sabattini[1], and Guillaume Sartoretti[2]

[1] University of Modena and Reggio Emilia, Department of Sciences and Methods for Engineering, Italy, {alessandro.bonetti, silvia.proia, lorenzo.sabattini}@unimore.it

[2] National University of Singapore, Department of Mechanical Engineering, Singapore, {zhouxin, yifeng, e1280621, e1192634}@u.nus.edu, guillaume.sartoretti@nus.edu.sg

[*] Corresponding author

Abstract. Autonomous Vehicles (AVs) operating in multi-agent urban environments face complex navigation challenges, as each must pursue its own objectives while continuously interacting with others. Recent Multi-Agent Reinforcement Learning (MARL) approaches have shown encouraging progress in cooperative decision-making, but they often lead to unsafe driving behaviors and lack modularity and interpretability. To address these limitations, we introduce MACH3, a fully decentralized Multi-Agent Cooperative Hierarchical Reinforcement Learning architecture, in which each agent operates solely based on local observations while enabling implicit coordination through learned policies. MACH3 decomposes AV navigation into three layers: a high-level cooperative MARL policy capturing global inter-vehicle interactions, a middle-level planner generating feasible trajectories, and a low-level controller responsible for accurate motion control. Experiments in urban driving scenarios demonstrate that MACH3 outperforms state-of-the-art methods in terms of success rate, performance, and driving comfort, while maintaining flexibility and robustness through its modular and decentralized design.

Keywords: Multi-Vehicle Autonomous Driving · Multi-Agent Reinforcement Learning · Multi-Agent System · Hierarchical Control.

1 Introduction

Autonomous Vehicles (AVs) have emerged as a popular topic in robotics and artificial intelligence research due to their potential to improve safety, enhance traffic efficiency, and support future transportation systems. A core challenge in the AV domain remains *navigation*, a hierarchical process that integrates decision making, motion planning, and vehicle control to ensure safe and efficient

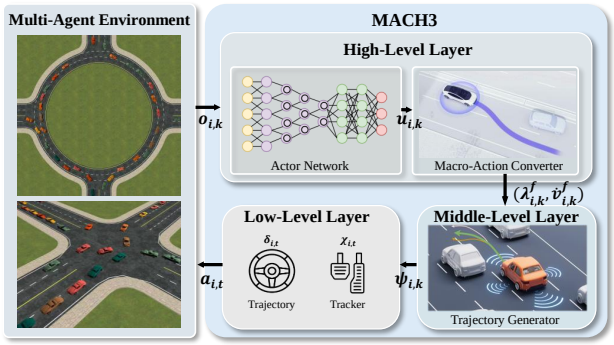

Fig. 1. Overview of MACH3. Each agent receives individual observations, and processes them through a hierarchical pipeline. The high-level layer selects a waypoint based on long-term objectives and cooperation with other vehicles, specifying target lane and desired speed; the middle-level layer generates a feasible trajectory; and the low-level layer ensures its tracking to produce the final control actions.

movement toward designated goals [1]. Traditional approaches, typically based on rule-based heuristics and predefined behavior strategies [2], offer high reliability in standardized scenarios, but struggle in dynamic traffic environments due to limited adaptability to unpredictable interactions. In contrast, Reinforcement Learning (RL) enables AVs to learn adaptive control policies directly from interaction data [3]. However, End-to-End (E2E) RL frameworks often lack interpretability and reliable, accurate control in multi-agent environments.

Although significant progress has been made in allowing individual vehicles to perceive their environment and navigate using single-agent RL, existing methods [4] fall short in Multi-Agent Self-Driving (MASD) scenarios [5], where multiple AVs must operate in a fully decentralized manner, relying only on local observations, and coordinate in complex urban road networks with multiple lanes (see Fig. 1). In MASD settings, the complexity of vehicle interactions grows dramatically, as each AV must not only pursue its own objectives, but also cooperate with other agents to ensure efficient traffic. Hence, researchers have turned to Multi-Agent Reinforcement Learning (MARL) [6], which has proven effective in fostering cooperation and coordination among agents. For instance, Coordinated Policy Optimization (CoPO) [5] introduces learnable local coordination factors combined with meta-gradient learning, but provides limited interpretability of inter-agent interactions. Traffic Coordinator Network (TraCo) [7] leverages cross-attention and counterfactual advantage functions to better estimate each agent's contribution to its neighborhood, yet remains limited to local interactions. Similarly, Socially-Attentive Policy Optimization (SAPO) [8] employs a self-attention mechanism to model pairwise interactions, but mainly focuses on the most relevant neighbor and fails to capture global dynamics among all agents.

To the best of the authors' knowledge, most learning-based MASD frameworks adopt E2E MARL architectures [9], mapping sensor data, ego-vehicle states, and navigation inputs directly to control commands while jointly handling decision-making and motion control. Despite promising results, E2E approaches [10] often produce low-quality driving behaviors, such as oscillatory

trajectories and jittery throttle-brake actions, and lack modularity and interpretability.

To address the above challenges, we propose a novel fully decentralized multi-agent framework grounded in Hierarchical Reinforcement Learning (HRL) [11], motivated by recent single-agent approaches, such as Continuous-Lattice (Cola)-HRL [12] and State-based Safety Enhancement via HRL (Safari) [13], which demonstrates its effectiveness in separating high-level decision-making from low-level control. We introduce MACH3, a Multi-Agent Cooperative Hierarchical RL architecture that overcomes the limitations of purely E2E MARL approaches by enabling both strategic adaptability and reliable motion control. MACH3 decomposes AV navigation into three layers: high-level cooperation, middle-level motion planning, and low-level motion tracking. Our high-level layer (HLL) employs a cooperative RL policy for strategic decisions, including waypoint selection and maneuver planning, while a Graph Attention Network (GAT) [14] captures global inter-vehicle dependencies to enhance coordination. The middle-level layer (MLL) relies on the sampling-based trajectory planner FRENETIX [15] to generate feasible and consistent trajectories aligned with high-level decisions. The low-level layer (LLL) ensures accurate execution through trajectory tracking. Results show that MACH3 outperforms state-of-the-art E2E MARL methods in task success rate, traffic performance, and driving comfort under complex scenarios. Our modular architecture improves interpretability and supports integration or modification of individual components, enhancing adaptability while preserving a fully decentralized execution paradigm.

2 Background

2.1 Multi-Agent Navigation Problem

We consider an urban multi-lane road network where multiple AVs navigate toward assigned destinations under decentralized control. Each agent selects driving behaviors (e.g., lane changes or speed adaptation) based on its state and local observations, including LiDAR and road-edge sensing. The goal is to minimize travel times, while maximizing success rate, i.e., reaching the destination safely by avoiding collisions, while also ensuring a smooth and comfortable ride.

We formulate the *multi-agent navigation problem* as a Macro-action Decentralized Partially Observable Markov Decision Process (MacDec-POMDP) [16]. Let $\mathcal{N} := \{1, \dots, N\}$ be the set of N agents. At each time step t, let us consider the subset of active agents indexed by $i \in \mathcal{N}_t := \{1, \dots, N_t\} \subseteq \mathcal{N}$, where N_t is the number of active agents, i.e., the AVs currently operating in the selected urban environment. Each agent $i \in \mathcal{N}_t$ enters the environment at its designated entry time step, denoted by t_i^{s}. Agents do not have direct access to the complete environment state $\mathcal{S}_t := \{s_{i,t}\}_{i \in \mathcal{N}_t} \in \mathcal{S}$, where $\mathcal{S}$ is the global state space and $s_{i,t} = [x_{i,t}, y_{i,t}, \vartheta_{i,t}, \dot{x}_{i,t}, \dot{y}_{i,t}, \dot{\vartheta}_{i,t}, \ddot{x}_{i,t}, \ddot{y}_{i,t}, \ddot{\vartheta}_{i,t}]^\top \in \mathbb{R}^9$ is the state of the agent $i \in \mathcal{N}_t$ active at time step t. Here, $x_{i,t}$, $y_{i,t}$, and $\vartheta_{i,t}$ denote the vehicle's linear and angular positions along the x–y axes, $\dot{x}_{i,t}$, $\dot{y}_{i,t}$, and $\dot{\vartheta}_{i,t}$ its linear and angular velocities, and $\ddot{x}_{i,t}$, $\ddot{y}_{i,t}$, and $\ddot{\vartheta}_{i,t}$ its linear and angular accelerations. All

quantities are expressed in the global frame, with the x- and y-axes aligned to the environment layout. Without access to $\mathcal{S}_t$, each agent $i \in \mathcal{N}_t$ instead receives a local observation $o_{i,t}$ through the observation function $o_{i,t} = \mathcal{Z}_i(\mathcal{S}_t) : \mathcal{S} \to \mathcal{O}$, with $\mathcal{O}$ the observation space.

Each agent $i \in \mathcal{N}_t$ selects a macro-action $u_{i,k_i} \in \mathcal{U}_i$ from the macro-action space $\mathcal{U}_i$ at the beginning of its k_i-th planning interval, where $k_i \in \{1, \dots, K_i\}$ and K_i is the total number of macro-actions executed while the agent is active. Each planning interval spans D consecutive time steps, over which the chosen macro-action u_{i,k_i} remains fixed. Specifically, at time step $t_{i,k_i} = k_i D + t_i^{\mathrm{s}}$, agent $i \in \mathcal{N}_t$ selects the macro-action u_{i,k_i} based on the current observation $o_{i,k_i} = o_{i,t}$, with $t = t_{i,k_i}$. The joint macro-action $\mathcal{U}_t := \{u_{i,k_i}\}_{i \in \mathcal{N}_t} \subset \mathcal{U}$ is defined as the set of macro-actions of the active agents at time step t, where $\mathcal{U}$ denotes the joint macro-action space. For the sake of notation clarity, we will omit the agent index i throughout the manuscript whenever its inclusion is not strictly necessary.

The macro-action $u_{i,k} \in \mathcal{U}_i$ for the active agent $i \in \mathcal{N}_t$ is performed over $t \in [t_{i,k},\, t_{i,k} + D - 1]$, throughout which a sequence of primitive actions is generated by a LLL policy: $a_{i,t} = f(u_{i,k}), \quad \forall t \in [t_{i,k},\, t_{i,k}+D-1]$. The execution of the joint action $\mathcal{A}_t := \{a_{i,t}\}_{i \in \mathcal{N}_t} \in \mathcal{A}$, where $\mathcal{A}$ denotes the joint primitive action space, induces a state transition via the transition function $\mathcal{P}(\mathcal{S}_{t+1} \mid \mathcal{S}_t, \mathcal{A}_t)$. Subsequently, each active agent $i \in \mathcal{N}_t$ receives an individual reward $r_{i,t}^{\mathrm{I}} = \mathcal{R}(\mathcal{S}_t, \mathcal{A}_t)$, with $\mathcal{R}$ representing the reward function. Finally, individual rewards are accumulated by each agent $i \in \mathcal{N}_t$ over its k-th planning interval as $R_{i,k}^{\mathrm{I}} = \sum_{t=t_{i,k}}^{t_{i,k}+D-1} r_{i,t}^{\mathrm{I}}$, corresponding to the selected macro-action $u_{i,k}$. Finally, ρ_0 and γ denote the initial state distribution and the discount factor, respectively.

With all components defined, we represent the MacDec-POMDP as the tuple $\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \mathcal{O}, \mathcal{Z}, \mathcal{U}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0, \gamma \rangle$.

Given the macro-control policy π_i, parameterized by θ_i, the individual objective for agent $i \in \mathcal{N}_t$ is defined as the expected discounted return $J_i^{\mathrm{I}}(\theta_i) = \mathbb{E}\Big[\sum_{k=1}^{K_i} \gamma^k R_{i,k}^{\mathrm{I}}\Big]$. The global objective is expressed as $J_i^{\mathrm{G}}(\theta_i) = \mathbb{E}\Big[\sum_{k=1}^{K_i} R_{i,k}^{G}\Big]$, where the global average reward is given by $R_{i,k}^{\mathrm{G}} = \sum_{t=t_{i,k}}^{t_{i,k}+D-1} \frac{\sum_{j \in \mathcal{N}_t} r_{j,t}^{\mathrm{I}}}{|\mathcal{N}_t|}$.

The overall objective for each agent $i \in \mathcal{N}_t$ is defined as: $J(\theta_i) = J_i^{\mathrm{I}}(\theta_i) + \omega^{\mathrm{G}} J_i^{\mathrm{G}}(\theta_i)$, where ω^{G} is a weighting factor that balances the contribution of the global objective relative to the individual objective.

2.2 FRENETIX Motion Planner

FRENETIX is a publicly available framework for generating kinematically feasible trajectories [15]. Its modular design and sampling strategy make it suitable for real-time applications. The framework operates in the Frenet coordinate frame, decoupling lateral and longitudinal motion with respect to a reference path Γ_i for each agent $i \in \mathcal{N}_t$. Based on Γ_i, the selected framework can transform a global Cartesian frame state $s_{i,t}$ into the corresponding Frenet frame state $\eta_{i,t} = F(s_{i,t})$ through the coordinate transformation F, defined in [15], and vice versa using $s_{i,t} = F^{-1}(\eta_{i,t})$. The Frenet state $\eta_{i,t}$ is defined as $\eta_{i,t} = [\nu_{i,t}, \dot{\nu}_{i,t}, \ddot{\nu}_{i,t}, d_{i,t}, \dot{d}_{i,t}, \ddot{d}_{i,t}]^\top \in \mathbb{R}^6$, where $\nu_{i,t}$, $\dot{\nu}_{i,t}$, and $\ddot{\nu}_{i,t}$ denote the

vehicle's longitudinal position, speed, and acceleration, and $d_{i,t}$, $\dot{d}_{i,t}$, and $\ddot{d}_{i,t}$ denote its lateral position, speed, and acceleration relative to the reference path Γ_i. Accordingly, FRENETIX computes the trajectory $\psi_{i,k} = \{(s^*_{i,t}, \eta^*_{i,t}) \mid t \in [t_{i,k}, \dots, t_{i,k} + H]\}$ in both Cartesian and Frenet frames over a planning horizon H, with $H > D$ to ensure sufficient look-ahead for decision-making and planning. The superscript * denotes the nominal trajectory states $s^*_{i,t}$ and $\eta^*_{i,t}$ employed for planning and model linearization.

To generate the trajectory $\psi_{i,k}$, the framework operates through a structured sequence of steps. Given the state $s_{i,t_{i,k}}$ of agent $i \in \mathcal{N}_t$, its reference path Γ_i, and a waypoint $(d^{\mathrm{f}}_{i,k}, \dot{\nu}^{\mathrm{f}}_{i,k})$, FRENETIX first generates a discrete lattice $\mathcal{G}_i^{\psi(0)}$ of $\Psi^{(0)}$ candidate trajectories $\psi^{(0)}_{i,p}$, with $p \in \{1, \dots, \Psi^{(0)}\}$. $\mathcal{G}_i^{\psi(0)}$ is constructed by combining a predefined number of N^t terminal time samples with a predefined number of longitudinal terminal position samples N^ν, resulting in a total of $\Psi^{(0)} = N^t N^\nu$ candidate trajectories. Specifically, each $\psi^{(0)}_{i,p}$ is uniquely defined by a terminal time and a longitudinal terminal position, which are sampled within their respective user-defined intervals, $[t^{\min}, t^{\max}]$ and $[\nu^{\min}, \nu^{\max}]$.

Each candidate trajectory $\psi^{(0)}_{i,p} \in \mathcal{G}_i^{\psi(0)}$ is then rigorously evaluated to ensure compliance with vehicle kinematic limitations and road boundaries constraints. The feasible trajectories, denoted as $\psi^{(1)}_{i,n}$ with $n \in \{1, \dots, \Psi^{(1)}\}$, are gathered into the subset $\mathcal{G}_i^{\psi(1)} \subseteq \mathcal{G}_i^{\psi(0)}$. Finally, a user-defined total cost function is computed for each feasible trajectory $\psi^{(1)}_{i,n} \in \mathcal{G}_i^{\psi(1)}$, and the trajectory with the lowest cost is selected as the optimal solution $\psi_{i,k}$.

3 Modular Architecture

To address the *multi-agent navigation problem*, we introduce MACH3, a decentralized framework based on a modular architecture composed of three primary layers: the HLL (see Section 3.1), the MLL (see Section 3.2), and the LLL (see Section 3.3), each handling a different stage from strategic planning to motion execution. The HLL is modeled as a MARL policy with an attention mechanism to enable cooperation and guide downstream decision-making. At the start of the k-th planning interval for agent $i \in \mathcal{N}_t$, the HLL receives the observation $o_{i,k}$ and generates a macro-action $u_{i,k}$, subsequently converted into a waypoint $(\lambda^{\mathrm{f}}_{i,k}, \dot{\nu}^{\mathrm{f}}_{i,k})$ specifying a desired lane $\lambda^{\mathrm{f}}_{i,k}$ and a target longitudinal speed $\dot{\nu}^{\mathrm{f}}_{i,k}$. The MLL uses the FRENETIX planner to generate a feasible trajectory $\psi_{i,k}$ connecting the current state $s_{i,t_{i,k}}$ to the waypoint. The LLL then tracks $\psi_{i,k}$ over $t \in [t_{i,k}, t_{i,k} + D - 1]$, producing control actions $a_{i,t} = (\delta_{i,t}, \chi_{i,t})$, corresponding to steering and throttle, ensuring accurate execution.

3.1 High-Level Layer

RL Components

Observation: Inspired by [17], we define the observation as follows. At the beginning of the k-th planning interval, the agent $i \in \mathcal{N}_t$ receives its observation

$o_{i,k} \in \mathbb{R}^{93}$, composed of three components: ego-state vector $o^{\mathrm{s}}_{i,k} \in \mathbb{R}^9$, navigation-related vector $o^{\mathrm{n}}_{i,k} \in \mathbb{R}^{12}$, and LiDAR-based perception vector $o^{\mathrm{l}}_{i,k} \in \mathbb{R}^{72}$. The ego-state vector $o^{\mathrm{s}}_{i,k}$ encodes the vehicle's current configuration, including lateral distances to road boundaries, relative heading to the road tangent, longitudinal and lateral velocities, steering angle, yaw rate, last executed macro-action, and lateral offset from the lane centerline. The navigation-related vector $o^{\mathrm{n}}_{i,k}$ provides road geometry and lane topology, e.g., the number of available lanes on each side, the curvature and relative orientation of the current and upcoming road segments. Finally, the LiDAR-based vector $o^{\mathrm{l}}_{i,k}$ captures the positions and relative distances of obstacles within the vehicle's field of view.

Macro-Action: A macro-action $u_{i,k} = [u^{\lambda}_{i,k}, u^{\nu}_{i,k}]^{\top} \in [0,1]^2 \subset \mathbb{R}^2$ consists of two dimensional continuous signals. This vector is subsequently transformed into the desired waypoint $(\lambda^{\mathrm{f}}_{i,k}, \dot{\nu}^{\mathrm{f}}_{i,k})$ that the ego vehicle aims to reach. Specifically, $u^{\lambda}_{i,k}$ is mapped to the selected lane index $\lambda^{\mathrm{f}}_{i,k}$ as follows:

$$\lambda^{\mathrm{f}}_{i,k} = \lambda^{\mathrm{c}}_{i,k} + \begin{cases} -1, & \text{if } u^{\lambda}_{i,k} < 1/3 \\ 0, & \text{if } 1/3 < u^{\lambda}_{i,k} < 2/3 \\ +1, & \text{if } u^{\lambda}_{i,k} > 2/3 \end{cases},$$

where $\lambda^{\mathrm{c}}_{i,k}$ is the index of the current lane occupied by the vehicle. Similarly, $u^{\nu}_{i,k}$ is mapped to the desired longitudinal speed $\dot{\nu}^{\mathrm{f}}_{i,k} = \dot{\nu}^{\max}\, u^{\nu}_{i,k}$ at the end of the trajectory generated by the MLL (as will be detailed in Section 3.2). Here, $\dot{\nu}^{\max}$ represents the maximum longitudinal speed.

Reward: Our individual reward function combines three dense components and a sparse terminal reward: $r^{\mathrm{I}}_{i,t} = c_1\, r^{\mathrm{d}}_{i,t} + c_2\, r^{\mathrm{s}}_{i,t} + c_3\, r^{\mathrm{l}}_{i,t} + r^{\mathrm{t}}$, where the driving reward is calculated as $r^{\mathrm{d}}_{i,t} = \nu_{i,t} - \nu_{i,t-1}$, with $\nu_{i,t}$ and $\nu_{i,t-1}$ denoting the longitudinal positions of the vehicle on the current lane λ^{c}_i at two consecutive time steps. The speed reward $r^{\mathrm{s}}_{i,t}$ is calculated by normalizing the current vehicle speed $|\frac{\dot{\nu}_{i,t}}{\dot{\nu}^{\max}}|$. Instead, the lateral reward $r^{\mathrm{l}}_{i,t} = -|d_{i,t} - d_{i,t-1}|$ is based on the absolute value of the lateral offset, penalizing lateral deviations, and stabilizing the lane selection behavior. The termination reward r^{t} comprises sparse components reflecting the episode outcome: $r^{\mathrm{succ}} = 10$ for successfully reaching the goal, and penalties $r^{\mathrm{out}} = -10$ for leaving the drivable area, $r^{\mathrm{way}} = -10$ for generating an unfeasible waypoint, and $r^{\mathrm{coll}} = -10$ for collisions. The total reward is then weighted using predefined factors $c_1 = c_2 = c_3 = 0.2$ following [17].

Cooperative MARL Architecture

The proposed cooperative Multi-Agent Reinforcement Learning (MARL) architecture extends Twin Delayed Deep Deterministic Policy Gradient (TD3) [18] under the centralized training with decentralized execution (CTDE) [19] paradigm by introducing a centralized global critic enhanced with a Graph Attention Network (GAT) to model inter-agent interactions. A local critic estimates individual Q-values from local observations, while the global critic captures cooperative behavior using privileged information. The actor is trained from both signals,

combining individual and cooperative objectives. Parameter sharing further improves training efficiency while enabling decentralized execution.

The training process relies on the experience buffer $\mathcal{B}$, which stores the transition tuples $b_{i,k} = (o_{i,k}, o_{i,k+1}, R^{\mathrm{I}}_{i,k}, R^{\mathrm{G}}_{i,k}, \mathcal{U}_k, \mathcal{U}_{k+1}, \mathcal{S}_k, \mathcal{S}_{k+1}, \zeta_{i,k})$ for the k-th planning interval of the agent $i \in \mathcal{N}_t$. Here, we denote the joint macro-action and global state at the beginning of the k-th planning interval as: $\mathcal{U}_k := \mathcal{U}_t, \quad \mathcal{S}_k := \mathcal{S}_t$, with $t = t_{i,k}$. $\zeta_{i,k}$ is a binary mask indicating whether agent $i \in \mathcal{N}_t$ is in a terminal state at the planning step k.

During training, transitions sampled from $\mathcal{B}$ are used to update both individual and global critic parameters θ. The local critics, parameterized by ϕ^{I}_1 and ϕ^{I}_2, follow the TD3 algorithm [20], which employs double Q-learning and delayed policy updates to mitigate overestimation bias. They are optimized by minimizing the MSE between estimated and target individual Q-values, calculated as:

$$L(\phi^{\mathrm{I}}_l, \mathcal{B}) = \mathop{\mathbb{E}}_{b_{i,k}\sim\mathcal{B}}\left[\left(Q_{\phi^{\mathrm{I}}_l}(o_{i,k}, u_{i,k}) - y^{\mathrm{I}}_{i,k}\right)^2\right] \quad \forall l \in \{1,2\}, \tag{1}$$

where $y^{\mathrm{I}}_{i,k}$ denotes the target individual value computed with the smoothed target policy and the minimum of the individual Q-values from the target local critic networks: $y^{\mathrm{I}}_{i,k} = R^{\mathrm{I}}_{i,k} + \gamma(1-\zeta_{i,k}) \min_{l=1,2} Q_{\phi^{\mathrm{I},\ \mathrm{targ}}_l}(o_{i,k+1}, u_{i,k+1})$.

The macro-action used to construct the Q-learning targets is generated by the target policy $\pi^{\theta^{\mathrm{targ}}}$ with additive noise $\epsilon \sim \mathrm{Normal}(0, \sigma)$, where σ is the standard deviation, and the noise is clipped within the range ς. The resulting action is computed as $u_{i,k+1} = \mathrm{clip}\Big(\pi^{\theta^{\mathrm{targ}}}(o_{i,k+1}) + \mathrm{clip}(\epsilon, -\varsigma, \varsigma), u^{\mathrm{low}}, u^{\mathrm{high}}\Big)$.

Similarly, the global centralized critics are trained by minimizing the MSE between estimated and target global Q-values:

$$L(\phi^{\mathrm{G}}_l, \mathcal{B}) = \mathop{\mathbb{E}}_{b_{i,k}\sim\mathcal{B}}\left[\left(Q_{\phi^{\mathrm{G}}_l}(o_{i,k}, \mathcal{U}_k, \mathcal{S}_k) - y^{\mathrm{G}}_{i,k}\right)^2\right] \ \forall l \in \{1,2\}, \tag{2}$$

where $y^{\mathrm{G}}_{i,k}$ is the target global value computed using the smoothed target policy and the minimum of the Q-values from the target centralized global critic networks, namely $y^{\mathrm{G}}_{i,k} = R^{\mathrm{G}}_{i,k} + \gamma(1-\zeta_{i,k}) \min_{l=1,2} Q_{\phi^{\mathrm{G},\ \mathrm{targ}}_l}(O_{i,k+1}, \mathcal{S}_{k+1}, \mathcal{U}_{k+1})$, where each element of the joint macro-action $\mathcal{U}_{k+1}$ is computed following the same procedure as in the local critic.

The actor network is trained to maximize the expected return by updating the policy to increase Q-values, jointly optimizing both local and global components:

$$L(\theta, \mathcal{B}) = -\mathop{\mathbb{E}}_{b_{i,k}\sim\mathcal{B}}\left[Q_{\phi^{\mathrm{I}}_1}(o_{i,k}, \pi^{\theta}(o_{i,k})) + \omega^{\mathrm{G}}\left(Q_{\phi^{\mathrm{G}}_1}(o_{i,k}, \pi^{\theta}(o_{i,k}), \mathcal{S}_k, \mathcal{U}_{\setminus i,k})\right)\right], \tag{3}$$

where $Q_{\phi^{\mathrm{I}}_1}(\cdot)$ and $Q_{\phi^{\mathrm{G}}_1}(\cdot)$ represent the individual navigation and the global cooperative objective, respectively. Here, $\mathcal{U}_{\setminus i,k}$ denotes the joint macro-action of all agents excluding agent $i \in \mathcal{N}_t$ at the beginning of its planning step k. The target networks of local and global critics, and the actor are updated according to $\theta^{\mathrm{targ}} \leftarrow (1-\tau)\,\theta^{\mathrm{targ}} + \tau\,\theta$, where τ controls the update rate.

The global critic network is employed to estimate each global Q-values $Q_{\phi^{\mathrm{G}}_l}$ and its targets, by integrating the joint state $\mathcal{S}_k$ and the joint macro action $\mathcal{U}_k$ to capture dependencies and complex interactions in multi-agent scenarios.

To this end, we include a GAT [14], a commonly adopted design in prior multi-agent interaction modeling [21], which allows the ego agent $i \in \mathcal{N}_t$ to focus on relevant active agents $\mathcal{N}_{i,t} \subseteq \mathcal{N}_t$, with $t = t_{i,k}$, aggregating their information to capture critical interactions for the decision-making process. The GAT models the system as a fully connected graph $\mathcal{G}_t = (\mathcal{N}_t, \mathcal{E}_t)$, where $\mathcal{N}_t$ is the set of nodes, while $\mathcal{E}_t \in \mathcal{N}_t \times \mathcal{N}_t$ is the set of pairs of distinct nodes called edges with cardinality E_t. Each node $m \in \mathcal{N}_t$ corresponds to an active agent and each edge $(m, n) \in \mathcal{E}_t$ is associated with an attention coefficient $\alpha_{m,n}$, with $n \in \mathcal{N}_t$.

To construct the GAT input, we first build the state–macro-action set $\mathcal{F} := \{s_{m,k} \parallel u_{m,k}\}_{m \in \mathcal{N}_t}$ by concatenating the joint state $\mathcal{S}_k$ and joint macro-action $\mathcal{U}_k$. Each element $s_{m,k} \parallel u_{m,k}$ is then transformed by a two-layer Multilayer Perceptron (MLP) with nonlinear activation functions into a higher-dimensional node feature $h_m^{(0)} = \mathrm{MLP}(s_{m,k} \parallel u_{m,k})$. The resulting feature set $\mathcal{H}^{(0)} := \{h_m^{(0)}\}_{m \in \mathcal{N}_t}$ serves as the input to the GAT. The GAT consists of two sequential attention layers, each computing attention coefficients and aggregating features along edges (m, n). In the first layer, the coefficient $\alpha_{m,n}^{(0)}$, indicating the relative importance of agent n's information to agent m, is computed as follows:

$$\alpha_{m,n}^{(0)} = \mathrm{softmax}_j\left(\mathrm{LeakyReLU}\left(q^\top \left[W h_m^{(0)} \,\|\, W h_n^{(0)}\right]\right)\right), \tag{4}$$

where W is a learnable weight matrix and q a learnable attention vector. The operator $\mathrm{softmax}_j(\cdot)$ normalizes the attention scores over all neighbors n of node m into a probability distribution, and $\mathrm{LeakyReLU}(\cdot)$ is a nonlinear activation with a small negative slope that prevents neuron inactivity [22]. Once obtained, the normalized attention coefficients are used to generate the updated feature vector $h_m^{(1)}$: $h_m^{(1)} = \xi(\sum_{n \in \mathcal{N}_t} \alpha_{m,n}^{(0)} W h_n^{(0)})$, where ξ is the ReLU function [22].

The same GAT computation process is applied to the second GAT layer, which takes $\mathcal{H}^{(1)} = \{h_m^{(1)}\}_{m \in \mathcal{N}_t}$ as input, computes and normalizes the attention scores $\alpha_{m,n}^{(1)}$, and aggregates the features from the other nodes to produce the set of output node features $\mathcal{H}^{(2)}$. From $\mathcal{H}^{(2)}$, the feature vector $h_i^{(2)}$ of the considered agent $i \in \mathcal{N}_t$ is extracted and employed to compute the target global Q-values as $Q_{\phi_l^{\mathrm{G,\ targ}}} = \mathrm{MLP}(h_i^{(2)} \,||\, \mathrm{MLP}(o_{i,k}))$, $l \in \{1, 2\}$.

3.2 Middle-Level Layer

To efficiently guide the agent $i \in \mathcal{N}_t$ to the waypoint $(\lambda_{i,k}^{\mathrm{f}}, \dot{\nu}_{i,k}^{\mathrm{f}})$ provided by the HLL, it is crucial to employ a trajectory planning strategy that computes a smooth and feasible trajectory $\psi_{i,k}$. Formally, $\psi_{i,k}$ connects the vehicle's initial state $s_i^{\mathrm{b}} = s_{i,t_{i,k}}$ to a final state s_i^{f} that satisfies the waypoint constraints, within a finite planning horizon H. Among the various trajectory planning approaches [9], we adopt the FRENETIX framework (see Section 2.2) within our MACH3 architecture due to its computational efficiency and its ability to generate feasible trajectories that comply with vehicle kinematic constraints.

To generate the trajectory $\psi_{i,k}$, FRENETIX takes as input the initial state s_i^{b}, the reference path Γ_i, and the target waypoint $(\lambda_{i,k}^{\mathrm{f}}, \dot{\nu}_{i,k}^{\mathrm{f}})$. In this work, Γ_i denotes the predefined reference centerline, i.e., the lane with index $\lceil \Lambda/2 \rceil$,

where Λ is the total number of lanes. To align with FRENETIX's waypoint representation, the desired lane index $\lambda^{\mathrm{f}}_{i,k}$ is converted to the corresponding lateral coordinate: $d^{\mathrm{f}}_{i,k} = \left(\lambda^{\mathrm{f}}_{i,k} - \lceil \frac{\Lambda}{2} \rceil\right) w^{\lambda}$, where w^{λ} denotes the lane width. During the sampling phase, we customize the interval $[\nu_i^{\min}, \nu_i^{\max}]$ by imposing $\nu_i^{\min} = \min\{\dot{\nu}^{\mathrm{f}}_{i,k}, \dot{\nu}^{\mathrm{b}}_{i}\} t^{\min}$ and $\nu_i^{\max} = \max\{\dot{\nu}^{\mathrm{f}}_{i,k}, \dot{\nu}^{\mathrm{b}}_{i}\} t^{\max}$. By constraining the bounds through the current speed $\dot{\nu}^{\mathrm{b}}_{i}$ and target speed $\dot{\nu}^{\mathrm{f}}_{i,k}$, the sampling range covers longitudinal distances reachable under uniform acceleration, while excluding infeasible trajectories.

Each candidate trajectory $\psi^{(0)}_{i,p} \in \mathcal{G}_i^{\psi(0)}$ is first evaluated for feasibility with respect to vehicle kinematics and road boundaries. Then, to assess the feasible trajectories $\psi^{(1)}_{i,n} \in \mathcal{G}_i^{\psi(1)}$, we introduce the cost function $J^{\mathrm{tot}}_{i,n} = \sum_{t=t_{i,k}}^{t_{i,k}+H} \left(\dot{\nu}_{i,n,t} - \dot{\nu}^{\mathrm{f}}_{i,k}\right)^2 + \sum_{t=t_{i,k}}^{t_{i,k}+H} \left(d_{i,n,t} - d^{\mathrm{f}}_{i,k}\right)^2$. The first term enforces convergence toward the desired longitudinal speed, while the second penalizes deviations from the target lateral position. As a result, the MLL generates trajectories that rapidly converge to the waypoint $(\lambda^{\mathrm{f}}_{i,k}, \dot{\nu}^{\mathrm{f}}_{i,k})$, ensuring alignment with HLL objectives and enabling correct interpretation by the higher-level system. Thus, for the k-th planning interval of agent $i \in \mathcal{N}_t$, the MLL outputs the minimum-cost trajectory $\psi_{i,k}$, consisting of states $s^*_{i,t}$ for $t \in [t_{i,k}, t_{i,k}+D-1]$, where $\psi_{i,k} = \arg\min_{\psi^{(1)}_{i,n} \in \mathcal{G}_i^{\psi(1)}} J^{\mathrm{tot}}_{i,n}$.

3.3 Low-Level Layer

Once the MLL computes the trajectory $\psi_{i,k}$ for vehicle $i \in \mathcal{N}_t$, the objective is to track $\psi_{i,k}$ as accurately as possible. To this end, we adopt a Linear Quadratic Regulator (LQR) [23], as it offers efficient solutions for linearized models, balancing tracking accuracy, simplicity, and computational cost. Given the trajectory $\psi_{i,k}$, the tracking problem consists of finding the optimal control vector $\{a_{i,t} | t \in [t_{i,k}, t_{i,k}+D-1]\}$, such that the tracking and the control errors are minimized. The system model is linearized around the nominal trajectory $(s^*_{i,t}, a^*_{i,t})$, where $a^*_{i,t} = (\delta^*_{i,t}, \chi^*_{i,t}) = (0, 0)$. Assuming that the prediction horizon and control horizon have the same length, denoted as $N^{\mathrm{p}} = t_{i,k}+D-1$, the optimization problem for the k-th planning interval is formulated by introducing the cost function $J^{N^{\mathrm{p}}} = (s_{i,N^{\mathrm{p}}} - s^*_{i,N^{\mathrm{p}}})^{\top} Q_{N^{\mathrm{p}}} (s_{i,N^{\mathrm{p}}} - s^*_{i,N^{\mathrm{p}}}) + \sum_{n=t_{i,k}}^{N^{\mathrm{p}}-1} [(s_{i,n} - s^*_{i,n})^{\top} Q_n (s_{i,n} - s^*_{i,n}) + (a_{i,n} - a^*_{i,n})^{\top} R_n (a_{i,n} - a^*_{i,n})]$, where $Q_{N^{\mathrm{p}}} \in \mathbb{R}^{9\times 9}$, $Q_n \in \mathbb{R}^{9\times 9}$, and $R_n \in \mathbb{R}^{2\times 2}$ are the final cost, the state cost, and input cost diagonal positive definite matrices to be tuned. Note that the three terms in $J^{N^{\mathrm{p}}}$ present the final state deviation, state deviation, and input size, respectively. Given the initial state $s_{i,t_{i,k}}$, let $\{a^{\mathrm{out}}_{i,n} | n \in [t_{i,k}, N^{\mathrm{p}}]\}$ be the control sequence that minimizes the quadratic cost function $J^{N^{\mathrm{p}}}$ subject to the car-like model [24].

By iteratively solving the minimization problem with the cost function $J^{N^{\mathrm{p}}}$, the state constraint [24], and the initial state $s_{i,t_{i,k}}$, we obtain the optimal control law $a_{i,n} = K_n(s_{i,n} - s^*_{i,n}) + a^*_{i,n}$, $\forall n \in [t_{i,k}, N^{\mathrm{p}}]$, and the corresponding $s_{i,n}$ value. The feedback gain $K_n \in \mathbb{R}^{2\times 9}$ is computed through the well-known Riccati difference equations [23]. The process is repeated until convergence. Tracking can also be performed using alternative methods depending on computational resources and desired performance. Model Predictive Control [25] handles non-

linear dynamics and complex constraints at the cost of higher computation, while RL enables data-driven strategies that enhance adaptability and robustness.

4 Numerical Experiments

4.1 Experimental Setup

In this section, we present the experimental setup adopted to evaluate the performance of MACH3 and to benchmark it against state-of-the-art frameworks. All experiments were conducted in the MetaDrive simulator [17] on an Ubuntu server with 128 GB RAM, an NVIDIA RTX 3090 GPU, and an AMD Ryzen 9 5950X processor. Each method was trained for at least 1 million steps over three runs with different random seeds (roughly 36 hours). Evaluation used 20 fixed seeds, separate from training, for a total of 60 test episodes per method.

Environment Scenarios: Two representative MASD scenarios reflecting realistic urban environments are considered (see Fig. 1). We use the roundabout and intersection settings from the MetaDrive simulator [17], both featuring $\Lambda = 2$ lanes with width $w^{\lambda} = 3.5$ m, capturing the highly interactive and safety-critical nature of AV coordination in common urban driving scenarios, while also covering complementary interaction patterns, including merging and diverging in roundabouts and conflict resolution at intersections. The roundabout features continuous traffic around a central island with multiple entries and exits from four directions. A total of $N_t = 40$ vehicles are initially deployed and re-spawned upon completing their routes to maintain traffic density, allowing evaluation of lane changes, merging, and splitting maneuvers. The four-way intersection has no traffic signals, supports bi-directional flows while prohibiting U-turns, and deploys $N_t = 30$ vehicles re-spawned at random entry points, enabling assessment of negotiation and social behaviors among agents. In both scenarios, the fleet consists of homogeneous vehicles with a maximum longitudinal speed of $\dot{\nu}^{\max} = 8.3$ m/s, corresponding to a typical urban speed limit, a steering angle bounded by $\delta^{\max} = 40°$, maximum tangential acceleration and deceleration of $3.06, \mathrm{m/s^2}$ and $-10.0, \mathrm{m/s^2}$, respectively, and a wheelbase of $L = 2.5, \mathrm{m}$, following standard MetaDrive settings and reflecting realistic passenger vehicle characteristics in urban environments.

MACH3 Parameters: MACH3 relies on several parameters, which are adopted from common practices in RL [20] and trajectory planning [15], and empirically validated to ensure robust performance. All modules operate with a time step of 100 ms (MetaDrive default). In the HLL, the planning interval is set to $D = 4$ time steps, allowing frequent updates of agent decisions while maintaining consistency across actions. The policy training weighting factor $\omega^{\mathrm{G}} = 0.6$ balances individual contributions with global objectives. The HLL training hyper-parameters include the discount factor set to $\gamma = 0.99$, and the target network updated at a rate of $\tau = 0.005$. Learning rates are fixed at 1×10^{-5} for the actor and 2×10^{-5} for the critic. During critic updates, noise ϵ with a standard deviation of $\sigma = 0.2$ is clipped within the range $\varsigma = 0.5$ and added to the target policy, with delayed policy updates occurring every 3 steps. Neural network hidden layers contain

256 units each, providing sufficient capacity to capture the complexity of high-level decision-making. In the MLL, the prediction horizon is set to $H = 30$ time steps to allow the planned trajectory to reach the HLL waypoints. The sampling strategy is defined with $N^t = 17$ terminal times and $N^\nu = 17$ terminal longitudinal positions, providing a sufficiently dense lattice to generate different candidate trajectories and improve solution quality. The terminal time interval is set between $t^{\min} = 4$ and $t^{\max} = 30$ time steps, balancing feasibility and responsiveness. In the LLL, an LQR generates low-level control actions with a cost balancing tracking and effort. The input cost matrix $R = \mathrm{diag}(0.01 I_2)$ penalizes both actuators equally, while the state cost matrix $Q = \mathrm{diag}(5I_2, 7I_7)$ assigns higher weight to the first two states, where I_n denote a n-dimensional identity matrix. The final cost equals Q, and the prediction horizon $N^{\mathrm{P}} = 4$ matches the planning interval, providing sufficient look-ahead.

Baselines: We benchmark MACH3 against six state-of-the-art baselines and one hierarchical variant, selected to cover representative independent, cooperative, and interaction-aware MARL approaches: (1) *Individual Proximal Policy Optimization (IPPO)* [26], which learns individual E2E navigation strategies; (2) *Cooperative PPO (CPPO)*, following the CTDE paradigm with a centralized critic to promote cooperation; (3) *CoPO* [5], enabling bi-level coordination via meta-gradient learning; (4) *Individual TD3 (ITD3)* [18], an individual deterministic actor-critic without inter-agent coordination; (5) *Mean Field PO (MFPO)* [5], which leverages mean-field theory to model multi-agent interactions; (6) *TraCo* [7], a cooperative MARL method for multi-agent trajectory coordination; and (7) *Hierarchical Individual TD3 (HRL-ITD3)*, a non-cooperative hierarchical baseline replacing the cooperative HLL policy with ITD3, removing global coordination while retaining the hierarchical structure. All baselines share the same observation, action, reward, and termination settings for fair comparison. Except for HRL-ITD3, which follows our training setup, the others use default configurations and are trained until full convergence.

Metrics: To evaluate the performance of all methods, ten episodic Key Performance Indicators (KPIs) are considered, grouped into three categories: *Rates*, *Performance*, and *Comfort*. Each category captures different aspects of AV behavior and system performance, offering a thorough evaluation of outcomes and driving quality, as commonly adopted in this type of analysis [12, 27, 17]. For the sake of clarity, we indicate for each KPI whether higher (↑) or lower (↓) values correspond to better performance.

The *Rates* category measures task completion and safety, including the *Success Rate (SR)* ↑, the fraction of vehicles that reach their destination; the *Off-Road Rate (OR)* ↓, the rate of AVs that leave the road; the *No-Valid-Trajectory Rate (WR)* ↓, the fraction of vehicles assigned an unreachable waypoint by the HLL; the *Timeout Rate (TR)* ↓, the rate of vehicles that fail to reach their destination within the episode time limit; and the *Collision Rate (CR)* ↓, the rate of AVs involved in collisions. The *Performance* category evaluates operational efficiency through *Efficiency (EFF)* ↑, defined as the normalized success–failure balance $(N^{\mathrm{succ}} - N^{\mathrm{fail}})/T^{\mathrm{ep}}$, where T^{ep} is the episode duration, and N^{succ} and N^{fail} de-

Table 1. Results of MACH3 and baselines. Values are reported as avg (std); best in **bold**, second-best underlined.

Method	Intersection environment with 30 initialized agents (No U-Turn)				
	SR ↑	OR ↓	WR ↓	TR ↓	CR ↓
IPPO	68.41 (7.22)	5.70 (2.34)	-	0.77 (2.07)	25.12 (6.64)
CPPO	61.10 (6.47)	8.36 (4.09)	-	0.12 (0.35)	30.66 (6.74)
CoPO	71.75 (9.48)	6.68 (3.64)	-	2.13 (5.05)	19.44 (7.21)
ITD3	69.26 (3.78)	6.60 (0.97)	-	0.11 (0.10)	24.04 (3.17)
MFPO	68.65 (4.44)	11.15 (3.34)	-	0.00 (0.00)	20.20 (4.44)
TraCo	63.75 (10.16)	12.18 (2.52)	-	0.68 (1.72)	23.39 (8.92)
HRL-ITD3	59.18 (6.05)	0.00 (0.00)	**2.98 (1.68)**	**0.00 (0.00)**	37.84 (6.12)
MACH3 (ours)	**75.61 (6.97)**	**0.00 (0.00)**	3.33 (2.03)	3.48 (5.01)	**17.57 (6.11)**
	EFF ↑	ATS ↓	Avg SCR ↓ $[\frac{rad}{s}]$	Avg Acc ↓ $[\frac{m}{s^2}]$	Avg ACR ↓ $[\frac{m}{s^3}]$
IPPO	32.4 (12.7)	639.88 (59.84)	4.51 (0.25)	3.68 (0.29)	26.66 (1.63)
CPPO	24.3 (15.5)	612.15 (55.49)	3.99 (0.45)	3.92 (0.45)	26.64 (2.79)
CoPO	38.7 (15.2)	613.69 (65.20)	4.12 (0.92)	3.38 (0.29)	23.71 (1.87)
ITD3	**51.7 (10.54)**	**492.92 (35.22)**	2.25 (0.09)	4.16 (0.27)	22.43 (1.29)
MFPO	37.7 (9.22)	574.23 (34.49)	3.68 (0.13)	4.09 (0.20)	28.39 (1.33)
TraCo	39.90 (26.82)	535.85 (84.68)	1.60 (0.23)	3.67 (0.68)	18.04 (3.20)
HRL-ITD3	24.4 (13.8)	617.40 (48.51)	0.04 (0.01)	2.13 (0.09)	8.61 (0.39)
MACH3 (ours)	43.9 (9.95)	625.94 (44.76)	**0.03 (0.01)**	**1.79 (0.19)**	**7.50 (0.85)**
Method	Roundabout environment with 40 initialized agents				
	SR ↑	OR ↓	WR ↓	TR ↓	CR ↓
IPPO	65.04 (15.36)	6.59 (3.72)	-	17.16 (15.38)	11.21 (4.96)
CPPO	71.87 (8.29)	7.60 (3.47)	-	1.13 (2.29)	19.39 (7.18)
CoPO	69.07 (9.68)	7.82 (3.08)	-	8.61 (7.45)	14.48 (7.05)
ITD3	83.28 (0.81)	4.67 (1.47)	-	0.37 (0.37)	11.69 (0.77)
MFPO	75.17 (7.50)	9.71 (2.81)	-	3.40 (2.84)	11.72 (4.39)
TraCo	69.21 (5.59)	20.76 (3.31)	-	2.08 (3.37)	7.99 (3.98)
HRL-ITD3	78.31 (6.21)	0.00 (0.00)	**2.29 (1.41)**	**0.00 (0.00)**	19.39 (6.14)
MACH3 (ours)	**88.80 (4.19)**	**0.00 (0.00)**	4.22 (2.41)	3.73 (1.58)	**3.24 (3.99)**
	EFF ↑	ATS ↓	Avg SCR ↓ $[\frac{rad}{s}]$	Avg Acc ↓ $[\frac{m}{s^2}]$	Avg ACR ↓ $[\frac{m}{s^3}]$
IPPO	39.1 (18.6)	737.47 (66.89)	3.55 (0.45)	4.30 (0.39)	27.63 (2.46)
CPPO	46.1 (16.4)	667.50 (46.55)	2.99 (0.43)	4.59 (0.35)	28.38 (2.22)
CoPO	40.9 (13.9)	727.46 (46.55)	3.92 (0.34)	4.66 (0.21)	29.86 (1.51)
ITD3	**72.4 (3.35)**	**586.41 (32.39)**	1.90 (0.27)	4.43 (0.27)	21.34 (2.67)
MFPO	49.75 (12.01)	684.00 (39.41)	3.11 (0.07)	4.57 (0.12)	28.40 (0.69)
TraCo	44.75 (11.02)	674.72 (47.67)	3.03 (0.17)	5.33 (0.36)	23.52 (1.22)
HRL-ITD3	63.8 (11.4)	609.02 (31.68)	0.07 (0.00)	2.32 (0.03)	9.00 (0.19)
MACH3 (ours)	70.5 (6.4)	661.99 (23.52)	**0.06 (0.00)**	**2.05 (0.04)**	**8.25 (0.20)**

note the numbers of vehicles that succeed or fail to reach their goals, and the *Average Travel Steps (ATS)* ↓, the mean number of steps required by successful vehicles to complete their routes. Finally, the *Comfort* category measures driving smoothness through the *Average Steering Change Rate (Avg SCR)* ↓, the mean steering change across vehicles; the *Average Acceleration (Avg Acc)* ↓, the mean cumulative acceleration during the episode; and the *Average Acceleration Change Rate (Avg ACR)* ↓, which captures abrupt variations in vehicle motion.

4.2 Results

This section presents a thorough evaluation of MACH3, including a comparative analysis against state-of-the-art baselines, a scalability analysis, an ablation study, and an assessment of real-world applicability.

Comparative Analysis: Table 1 presents the comparative evaluation of MACH3 and baseline methods across the two traffic environments.

First, regarding *Rates*, MACH3 achieves the highest *SR*, reaching 75.61% at the intersection and 88.80% in the roundabout, outperforming all baselines and highlighting the benefit of our cooperative HLL compared to HRL-ITD3. MACH3

also attains an *OR* of 0%, demonstrating the MLL's effectiveness in enforcing road-boundary compliance. MACH3's *WR*, specific to hierarchical methods, remains low (3.33% at the intersection and 4.22% in the roundabout), confirming the reliability of the HLL-MLL structure in producing feasible waypoints and trajectories. HRL-ITD3 slightly outperforms MACH3 in terms of *WR*, likely due to its selfish policy that generates simple, easily executable waypoints for the LLL, but leads to higher collision risk. In contrast, MACH3 prioritizes global coordination through more complex and altruistic macro-actions, such as evasive lane changes, which approach kinematic limits and are more often rejected as infeasible, increasing *WR*, while significantly reducing the *CR*. MACH3 also exhibits a slightly higher *TR*, particularly at the intersection (3.48%), as cautious high-level decisions favor safety over speed, occasionally prolonging episodes. Baseline methods instead fail primarily through collisions driven by self-interested behaviors (e.g., CPPO: $CR = 30.66\%$, HRL-ITD3: $CR = 37.84\%$ at the intersection), whereas MACH3 trades minor conservative failures (higher *WR* and *TR*) for the lowest *CR* across both scenarios.

Regarding *Performance*, MACH3 exhibits slightly higher *ATS* than the baselines, likely because *ATS* is calculated only for vehicles that reach their destination, while MACH3's higher SR results in longer average travel times. Nevertheless, we note that the *EFF* metric remains the second highest in both scenarios, showing that, despite slightly longer per-agent times, MACH3 maintains superior overall throughput.

Finally, regarding *Comfort*, MACH3 achieves the lowest *Avg SCR* and *Avg ACR* among all methods, indicating fewer abrupt steering and acceleration changes, while the minimized *Avg Acc* reflects smoother overall vehicle dynamics. Along with the HRL-ITD3 results, they show that our hierarchical architecture contributes to improved ride comfort, a key factor for real-world AV deployment.

Scalability Analysis: To assess zero-shot generalization, we evaluate the methods by scaling the initial number of agents from 6 to 36 (intersection) and from 8 to 48 (roundabout) without retraining. As shown in Fig. 2, MACH3 consistently achieves the highest *SR*. While the baselines degrade sharply under heavy congestion, MACH3 shows a more gradual decline, maintaining a 77% *SR* even at maximum density in the roundabout scenario. This behavior indicates that MACH3 learns robust strategies across varying traffic densities, demonstrating improved zero-shot generalization to more complex environments.

Ablation Study: While the comparison with the HRL-ITD3 baseline highlights the necessity of the cooperative framework, an ablation study is conducted to analyze the contribution of MACH3's key components. The study is performed in the intersection scenario, as shown in Table 2. Removing the GAT module (MACH3 w/o GAT) reduces *SR* from 75.61% to 71.68% and significantly increases *CR*, confirming its role in improving interaction modeling and coordination. The cooperative weight ω^G is also analyzed: setting $\omega^G = 0$ (selfish) leads to frequent deadlocks and lower *SR*, while $\omega^G = 1$ (fully cooperative) reduces collisions but decreases *EFF*, lowering *SR* to 73.34% due to timeouts. A value of $\omega^G = 0.6$ achieves the best trade-off, effectively balancing safety and efficiency.

Table 2. Results of the ablation study. Values are reported as avg (std) over 10 KPIs; best in **bold**, second-best underlined.

Method	Intersection environment with 30 initialized agents (No U-Turn)				
	SR ↑	OR ↓	WR ↓	TR ↓	CR ↓
MACH3 w/o GAT	71.68 (5.14)	**0.00 (0.00)**	6.11 (2.29)	**0.00 (0.00)**	22.26 (4.58)
$\omega^G = 0$	67.54 (5.90)	**0.00 (0.00)**	6.42 (2.01)	0.21 (0.93)	25.88 (6.06)
$\omega^G = 1$	73.34 (6.64)	**0.00 (0.00)**	5.02 (2.14)	4.68 (5.63)	**16.96 (4.74)**
$\omega^G = 0.6$	**75.61 (6.97)**	**0.00 (0.00)**	**3.33 (2.03)**	3.48 (5.01)	17.57 (6.11)
	EFF ↑	ATS ↓	Avg SCR ↓ $[\frac{rad}{s}]$	Avg Acc ↓ $[\frac{m}{s^2}]$	Avg ACR ↓ $[\frac{m}{s^3}]$
MACH3 w/o GAT	**44.85 (9.23)**	**598.79 (45.14)**	0.06 (0.01)	2.06 (0.16)	8.55 (0.63)
$\omega^G = 0$	38.80 (11.21)	612.41 (45.60)	0.06 (0.01)	1.82 (0.19)	7.86 (0.81)
$\omega^G = 1$	42.50 (10.46)	640.50 (57.05)	0.05 (0.01)	**1.67 (0.23)**	**7.21 (0.97)**
$\omega^G = 0.6$	43.90 (9.95)	625.94 (44.76)	**0.03 (0.01)**	1.79 (0.19)	7.50 (0.85)

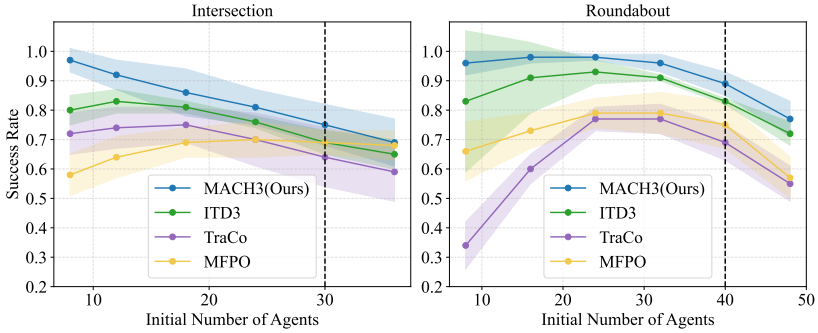

Fig. 2. Success rate and zero-shot generalization across different numbers of agents. All methods are trained at the default agent count (black dashed line).

Real-World Applicability: Real-time feasibility of MACH3 was evaluated on a laptop (Intel i7-12700H CPU, 16GB RAM, NVIDIA RTX 3050 GPU), representative of typical onboard AV deployment. Computation times were measured over 10,358 intervals, yielding average latencies of 0.3 ms, 10.2 ms, and 2.2 ms for the HLL, MLL, and LLL, respectively, with minimum values of 0.1 ms, 7.5 ms, and 0.8 ms, and maximum values of 0.7 ms, 32.1 ms, and 6.1 ms. All modules operate within real-time constraints under the adopted time step, confirming MACH3's suitability for real-time MASD scenarios. Furthermore, we experimentally validated MACH3 on a physical multi-robot platform, with demonstration videos provided in the supplementary materials.

5 Conclusions

This paper presents MACH3, a fully decentralized Multi-Agent Cooperative Hierarchical RL architecture for MASD navigation, composed of a high-level cooperative policy, a middle-level planner, and a low-level controller. Experiments show that MACH3 outperforms baselines in success rate, performance, and driving comfort, while reducing collisions and maintaining robust behavior under increasing traffic density. Real-time feasibility and validation on a physical multi-robot platform further demonstrate its applicability in realistic scenarios.

Future work will integrate an RL-based trajectory tracker in the low-level layer and incorporate emergency braking to further enhance safety.

Bibliography

[1] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intell. Veh.*, 1(1):33–55, 2016.

[2] Arne Kesting, Martin Treiber, and Dirk Helbing. General lane-changing model mobil for car-following models. *Transp. Res. Rec.*, 1999(1):86–94, 2007.

[3] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *J. Artif*, 4:237–285, 1996.

[4] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 Int. Conf. Robot. Autom. (ICRA)*, pages 8248–8254, 2019.

[5] Zhenghao Peng, Quanyi Li, Ka Ming Hui, Chunxiao Liu, and Bolei Zhou. Learning to simulate self-driven particles system with coordinated policy optimization. *Adv. Neural Inf. Process. Syst*, 34:10784–10797, 2021.

[6] Ruihua Han, Shengduo Chen, and Qi Hao. Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning. In *2020 IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 448–454, 2020.

[7] Weiwei Liu, Wei Jing, lingping Gao, Ke Guo, Gang Xu, and Yong Liu. Traco: Learning virtual traffic coordinator for cooperation with multi-agent reinforcement learning. In *7th CoRL*, 2023.

[8] Zipeng Dai, Tianze Zhou, Kun Shao, David Henry Mguni, Bin Wang, and Jianye HAO. Socially-attentive policy optimization in multi-agent self-driving system. In *6th CoRL*, 2022.

[9] Siyu Teng, Xuemin Hu, Peng Deng, Bai Li, Yuchen Li, Yunfeng Ai, Dongsheng Yang, Lingxi Li, Zhe Xuanyuan, Fenghua Zhu, et al. Motion planning for autonomous driving: The state of the art and future perspectives. *IEEE Trans. Intell. Veh.*, 8(6):3692–3711, 2023.

[10] Yifeng Zhang, Jieming Chen, Tingguang Zhou, Tanishq Duhan, Jianghong Dong, Yuhong Cao, and Guillaume Sartoretti. Coin: Collaborative interaction-aware multi-agent reinforcement learning for self-driving systems. *arXiv preprint arXiv:2603.24931*, 2026.

[11] Manikandan Ganesan, Sivanathan Kandhasamy, Bharatiraja Chokkalingam, and Lucian Mihet-Popa. A comprehensive review on deep learning-based motion planning and end-to-end learning for self-driving vehicle. *IEEE Access*, 12:66031–66067, 2024.

[12] Lingping Gao, Ziqing Gu, Cong Qiu, Lanxin Lei, Shengbo Eben Li, Sifa Zheng, Wei Jing, and Junbo Chen. Cola-hrl: Continuous-lattice hierarchical reinforcement learning for autonomous driving. In *2022 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pages 13143–13150, 2022.

[13] Ziqing Gu, Lingping Gao, Haitong Ma, Shengbo Eben Li, Sifa Zheng, Wei Jing, and Junbo Chen. Safe-state enhancement method for autonomous driving via direct hierarchical reinforcement learning. *IEEE Trans. Intell. Transp. Syst*, 24(9):9966–9983, 2023.

[14] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[15] Rainer Trauth, Korbinian Moller, Gerald Würsching, and Johannes Betz. Frenetix: A high-performance and modular motion planning framework for autonomous driving. *IEEE Access*, pages 1–1, 2024.

[16] Miao Liu, Kavinayan Sivakumar, Shayegan Omidshafiei, Christopher Amato, and Jonathan P How. Learning for multi-robot cooperation in partially observable stochastic environments with macro-actions. In *2017 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pages 1853–1860. IEEE, 2017.

[17] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Trans. Pattern Anal. Mach. Intell*, 45(3):3461–3475, 2022.

[18] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[19] Christopher Amato. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2409.03052*, 2024.

[20] Stephen Dankwa and Wenfeng Zheng. Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. In *3rd ICVISP*, pages 1–5, 2019.

[21] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *Int. Conf. Mach. Learn.*, pages 2961–2970. PMLR, 2019.

[22] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[23] B.D.O. Anderson and J.B. Moore. *Optimal Control: Linear Quadratic Methods*. Dover Publications, 2007.

[24] Alessandro De Luca, Giuseppe Oriolo, and Claude Samson. Feedback control of a nonholonomic car-like robot. In *Robot. Motion Plan. Control*, pages 171–253. Springer, 2005.

[25] Tianqi Qie, Weida Wang, Chao Yang, Ying Li, Yuhang Zhang, Wenjie Liu, and Changle Xiang. An improved model predictive control-based trajectory planning method for automated driving vehicles under uncertainty environments. *IEEE Trans. Intell. Transp. Syst*, 24(4):3999–4015, 2023.

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

[27] Zhiyu Huang, Haochen Liu, Jingda Wu, and Chen Lv. Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving. *IEEE Trans. Neural Netw. Learn. Syst.*, 2023.