# NAVIVLA: SOCIALLY-AWARE ROBOT NAVIGATION VISION-LANGUAGE ACTION MODEL

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Socially-aware robot navigation remains meaningful yet challenging, as robots must jointly optimize goal reaching, collision avoidance, human comfort, and implicit social norms under uncertain environmental dynamics. Existing methods often depend on simplified 2D geometric states, while recent many VLA models are not directly designed for socially compliant navigation. We propose NaviVLA, an RL fine-tuned social navigation VLA planner equipped with a flow matching action expert. NaviVLA integrates BEV maps, multi-sensor input, and world model powered language context to perform social navigation reasoning. Moreover, to support scalable and realistic training, we further construct an IsaacSim-based social navigation benchmark with diverse pedestrian dynamics simulators, standardized baselines, and multiple metrics. Its crowd module integrates both rule-based models and diffusion-powered simulators trained on real-world crowd trajectory datasets. Additionally, we introduce a RLHF-based social norm encoder to align the policy with respect to human preferences. Experiments in simulation and a physical robot show that NaviVLA improves navigation success, safety, and social compliance, establishing a new VLA-based framework for socially-aware robot navigation.

*All models are wrong, but some are useful.*

George E. P. Box

## 1 INTRODUCTION

Socially aware robot navigation requires a robot to reach its goal while safely and naturally interacting with humans in shared spaces Che et al. (2020); Xie & Dames (2023); Dobrevski & Skočaj (2024); Kretzschmar et al. (2016b); Mavrogiannis & Knepper (2019), as illustrated in Fig. 1. This task is naturally suited to reinforcement learning (RL), since the robot must continuously adapt its actions through trial-and-error interactions with dynamic environments while optimizing navigation efficiency, safety, and social compliance. Existing RL-based planners usually learn navigation policies in simplified 2D simulators without embodied 3D sensing or realistic visual observations Chen et al. (2017); Everett et al. (2018); Chen et al. (2019); Liu et al. (2021); Wang et al. (2023; 2025b), which limits their generalization to real-world embodied scenarios. In contrast, vision language action (VLA) models provide a promising paradigm for social navigation by integrating multimodal perception with high-level human-robot interaction (HRI) reasoning for embodied decision making.

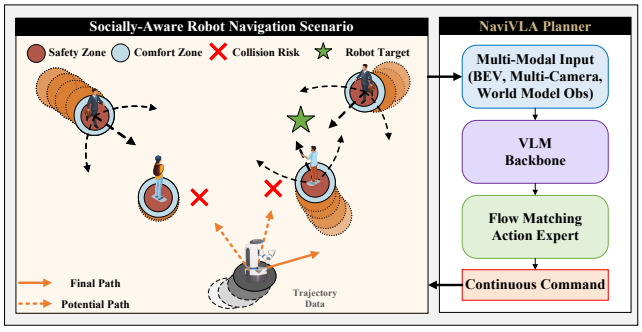



Figure 1: An illustration of NaviVLA in social navigation task.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

However, most existing VLA systems are primarily designed for manipulation tasks Zitkovich et al. (2023); Kim et al. (2025); Team et al. (2024); Black et al. (2024; 2025), rather than dense social navigation. Moreover, existing VLA navigation methods Cheng et al. (2025); Zhang et al. (2025; 2024) mainly target static navigation tasks without explicitly modeling pedestrian interactions, and they can not be fine-tuned through RL. In parallel, the VLA-based social navigation pipelines improve high-level social reasoning, but they typically use the VLA or VLM as a scoring, evaluation, or decision-support module, rather than as the direct decision-making policy Song et al. (2024); Munje et al. (2025); Payandeh et al. (2025). Therefore, how to adapt VLA into this social navigation task is still not studied well.

Moreover, despite recent embodied intelligence simulators provide realistic visual, or physical supports Xia et al. (2018); Shen et al. (2021); Savva et al. (2019); NVIDIA (2026); Biswas et al. (2022); Pérez-Higueras et al. (2023); Shcherbyna et al. (2025), they are not specifically designed for social navigation with realistic crowd generation and social compliance evaluation. To address this gap, we develop an IsaacSim-based social navigation benchmark with multi-sensor configurations, diverse evaluation metrics, and a diffusion-based crowd model trained on real-world trajectory datasets Bae et al. (2025); Lerner et al. (2007); Robicquet et al. (2016); Zhou et al. (2012); Majecka (2009).

The main contributions of this work are summarized as follows: (1) We propose **NaviVLA**, a unified framework for VLA-based socially-aware robot navigation. NaviVLA bridges multimodal robot perception to embodied intelligence decision-making by integrating multi-camera observations, bird's-eye-view (BEV) representations, and world-model-powered contextual information within an RL fine-tuned VLA policy equipped with a flow matching action expert Lipman et al. (2023). Beyond conventional distance-aware collision avoidance, this framework provides a foundation for robots to interpret richer contextual information and human requirements and translate them into socially appropriate navigation behaviors; (2) We develop a new IsaacSim-based social navigation benchmark with diverse scenarios, standardized baselines, and specific evaluation metrics. The benchmark supports multiple pedestrian models, including ORCA, social force, and a crowd diffusion network Bae et al. (2025), enabling systematic and fair evaluation of navigation policies under both rule-based and data-driven crowd dynamics; (3) We introduce a reinforcement learning from human feedback (RLHF)-based social norm encoder Rafailov et al. (2023) to align navigation behaviors with human preferences beyond rewards.

## 2 Literature Review

Social robot navigation is a foundational task in embodied intelligence and robotics, as it underpins further advanced robotic operations and enables robots to be deployed smoothly in human-shared public spaces. Early sample-based or optimization-based path planning approaches Hart et al. (1968); Karaman & Frazzoli (2011); Kavraki et al. (1998); Kallmann (2005); Banerjee & Chandrasekaran (2013) treat humans as static obstacles, which limits their effectiveness in highly dynamic environments. Thus, subsequent navigation pipelines explicitly consider moving agents and interaction dynamics, including VO-based (velocity obstacle), physics-informed, and geometric-based approaches Alonso-Mora et al. (2013); Helbing & Molnar (1995); Van den Berg et al. (2008); Rösmann et al. (2013). However, these methods still rely heavily on hand-crafted interaction patterns and often struggle to capture various of social norms or latent crowd interactions.

Recently, decoupled prediction-planning frameworks Cao et al. (2019); Du Toit & Burdick (2011); Kuderer et al. (2012); Katyal et al. (2020); Wang et al. (2022) have been widely adopted to mitigate the limitations of hand-crafted interaction models. These methods typically separate HRI inference from planning procedure, predicting human intentions before generating actions based on the inferred interaction dynamics. However, such decoupling may introduce failure cases such as the freezing robot problem Trautman et al. (2015) or reciprocal dance phenomena Johnson (2020), especially in dense crowds or spaces with limited collision-free regions. From a probabilistic perspective, exact inference in general Bayesian belief networks is NP-hard Cooper (1990). Therefore, fully joint long-horizon reasoning over human intentions, future trajectories, and robot actions remains computationally challenging.

Alternatively, coupled prediction-planning methods have been developed to approximate the mutual influence between robots and humans, enabling the robot to leverage HRI understanding during action execution. Recent coupled approaches Mavrogiannis & Knepper (2019); Trautman et al. (2015); Turnwald & Wollherr (2019); Mavrogiannis et al. (2022); Kretzschmar et al. (2016a); Samavi et al. (2024); Muchen Sun et al. (2025) abstract such interactions through game theory, topology, Gaus-

sian processes, and inverse reinforcement learning, etc. Apart from that, learning-based coupled frameworks Chen et al. (2017; 2019); Everett et al. (2018); Liu et al. (2021); Wang et al. (2023; 2025b); Chen et al. (2020); Wang et al. (2024; 2025a) benefit from the interactive nature of RL paradigm, where the navigation policy is optimized through closed-loop rollouts between the agent and environment. Notably, several methods Wang et al. (2023; 2025b); Sun et al. (2021) further encode pedestrian preference distributions using variational analysis or RLHF Lee et al. (2021a;b).

Recent advances in VLA models for robotics Zitkovich et al. (2023); Kim et al. (2025); Intelligence et al. (2025) demonstrate strong potential for embodied intelligence. Navigation-oriented VLA methods have expanded VLA models to embodied navigation, with some approaches introducing vision language reasoning for socially aware navigation guidance Cheng et al. (2025); Zhang et al. (2025; 2024); Payandeh et al. (2025); Narasimhan et al. (2025); Luo et al. (2025); Wen et al. (2025). However, these methods are either not optimized through RL or are not specifically designed for dense, human-populated environments.

# 3 METHODOLOGY

## 3.1 SOCIAL ROBOT NAVIGATION BENCHMARK

**(a). Task Definition:** As shown in Fig. 2, we formulate the social navigation task as a Dec-POMDP Åström (1965), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{U}, \mathcal{P}, \mathcal{S}_0, \mathcal{O}, \Omega, \mathcal{R}, \gamma, T \rangle$. The full environmental state is defined as $s_t = [s_t^r, s_t^{h_1}, \ldots, s_t^{h_N}, s_t^s] \in \mathcal{S}$, where $s_t^s$ denotes the static scene layout and environment context. The robot observation state is $s_t^{obs} = [s_t^r, o_t^{h_1}, \cdots, o_t^{h_N}, o_t^s] \sim \Omega$, where robot state is $s_t^r = [p_x, p_y, g_x, g_y, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]$, involving robot location, goal position, and the linear velocity and angular velocity of robot. Each pedestrian observable state is composed by the personal position and velocity as $s_t^{h_i} = [p_x^{h_i}, p_y^{h_i}, v_x^{h_i}, v_y^{h_i}]$, where $a_t^{h_i} = [v_x^{h_i}, v_y^{h_i}]$ denotes robot action.

Given the robot observation $o_t^r$, NaviVLA adopts a hierarchical control structure consisting of a high-level path planner and a low-level locomotion controller. The high-level planner $\pi_h$ generates a body-frame velocity command $a_t^r \sim \pi_h(a_t^r | o_t^r) = [v_x^{\text{cmd}}, v_y^{\text{cmd}}, \omega_z^{\text{cmd}}] \in \mathcal{A}$. This velocity command is tracked by the low-level locomotion RL-pretrained policy $\pi_l$, whose output is represented as joint actuation commands of robot legs $u_t^r \sim \pi_l(u_t^r | a_t^r) = [\dot{q}_t^1, \cdots, \dot{q}_t^{12}] \in \mathcal{U}$. The generated joint-pose commands are applied in the physics simulator, producing the next observation via the transition model $\mathcal{P}(s_{t+1} | s_t, u_t^r, a_t^{h_1}, \ldots, a_t^{h_N})$ with respect to IsaacSim physics, the robot locomotion controller, and the pedestrian simulator. The initial state distribution $\mathcal{S}_0$ samples robot initial states, goal positions, pedestrian positions, pedestrian goals, and scene layouts.

The robot observation is $o_t^r = [s_t^{obs}, I_t, B_t, L_t] \in \mathcal{O}$. Where $I_t$ denotes robot multi-camera images, and BEV map $B_t$ encodes the robot, goal, pedestrians, and local spatial layout. The language textual observation is generated by a social robot world model Wang et al. (2026) as $L_t = \text{WM}(s_t^{obs})$. This world model block converts the robot observations into structural textual descriptions with basic state information and semantic spatial-temporal HRI dynamics. The objective of Dec-POMDP is to learn a policy $\pi_h$ that can maximize the expected discounted cumulative reward $\mathcal{J}(\pi_h) = \mathbb{E}_{\pi_h, \mathcal{P}}[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t^r)]$, where $\mathcal{R}$ is the reward function with the discount factor $\gamma$ in horizon $T$.
**(b). Crowd Module:** The benchmark owns five human generation modes, involving *Human-Diffusion*, *Circle*, *Random*, *Corridor*, *Group*. The *Human-Diffusion* generator Bae et al. (2025) employs a pretrained diffusion network to approximate realistic pedestrian distribution and motion patterns. The *Circle* generator places pedestrians around a circle and assigns goals on the opposite side, creating dense crossing interactions with robot. The *Random* generator samples pedestrian start and goal positions randomly in the navigation area. The *Corridor* generator creates two-way pedestrian flow in a narrow passage. The *Group* generator initializes pedestrians in several groups. For each pedestrian $h_i$, the generator provides an initial position, velocity, goal, and policy preference as $\hat{s}_0^{h_i} = [p_{x,0}^{h_i}, p_{y,0}^{h_i}, v_{x,0}^{h_i}, v_{y,0}^{h_i}, g_x^{h_i}, g_y^{h_i}, v_{pref}^{h_i}]$. For variable size crowds in *Human-Diffusion*, a valid mask is used to indicate which pedestrian slots are active in the current episode.

The benchmark offers three pedestrian simulation modes to obtain human actions, including *Human-Diffusion*, ORCA, and Social Force. In the *Human-Diffusion* simulator Bae et al. (2025), pedestrian motion is generated to models realistic multi-agent crowd evolution conditioned on the current crowd state and scene context. In the *ORCA* simulator Alonso-Mora et al. (2013), each pedestrian selects a geometrically constrained collision-free velocity toward its goal while avoiding nearby pedestrians. In the *Social-Force* simulator Helbing & Molnar (1995), pedestrian motion is produced

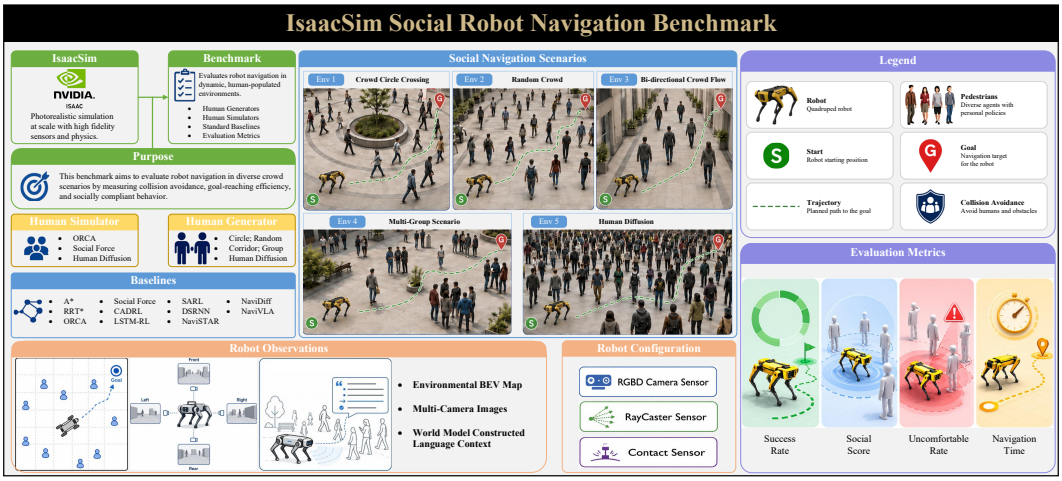



Figure 2: Overview of the IsaacSim Social Navigation Benchmark: The benchmark integrates diverse human simulators, evaluation metrics, and scene layouts such as room, empty room, and corridor. It supports multiple robots, including Spot, Anymal, and Go2, equipped with camera, RayCaster, and contact sensors.

by combining goal-directed attraction and interaction-based repulsive forces. The benchmark supports two types of pedestrian models that are IRA-based IssacSim pedestrian models and simplified cylinder models.

**(c). Robot Configuration:** The robot policy follows a hierarchical navigation stack with a high-level path planner $\pi_h$ and a low-level motion planner $\pi_l$. In NaviVLA, the high-level planner is implemented by the VLA module together with a flow matching action expert, while the low-level planner is an RL-pretrained motion controller. The benchmark provides several quadruped robot platforms, including Spot, Anymal, and Go2, with multiple sensor modalities, such as camera sensors, RayCaster sensors, and contact sensors.

**(d). Standardized Baseline:** The benchmark provides a standardized baseline set covering traditional approaches A* Hart et al. (1968), RRT* Karaman & Frazzoli (2011), Social Force Helbing & Molnar (1995), and ORCA Alonso-Mora et al. (2013), learning-based methods MLP and Transformer policies, and social navigation planners CADRL Chen et al. (2017), LSTM-RL Everett et al. (2018), SARL Chen et al. (2019), DSRNN Liu et al. (2021), NaviSTAR Wang et al. (2023), and NaviDIFF Wang et al. (2025b). For fair comparison, we re-implement all trainable baselines under the identical PPO Schulman et al. (2017) configurations and parameters.

**(d). Evaluation Metric:** We evaluate each policy from four perspectives: navigation efficiency, safety, social comfort, and social-norm compliance. The basic navigation metrics include success rate $SR$, average navigation time $NT$, average path length $NP$, and average velocity. Safety is measured by robot-human collision frequency and the minimum human distance. Social comfort is measured by the uncomfortable interaction rate. To provide a unified evaluation, we define a social navigation score $SC$ that jointly considers path quality, safety, and social norm satisfaction.

### 3.2 NaviVLA Framework

**(a). Overview:** As shown in Fig. 3, we propose NaviVLA, an RL and RLHF fine-tuned VLA framework for socially aware robot navigation. NaviVLA builds a navigation policy with respect to the multimodal observation input on top of a VLA model, and couples it with a flow matching action expert for continuous command generation. The objective of fine-tuning procedure is to learn a high-level navigation policy $\pi_h$ that guides the robot to its goal efficiently while avoiding collisions, preserving human comfort, and satisfying implicit social norms. As defined in the Dec-POMDP formulation, NaviVLA receives partial multimodal observations, including multi-camera images, BEV maps, and world model generated language descriptions. These modalities provide complementary information about local geometry, pedestrian motion, scene context, and social interaction relations. Then the inputs are encoded by a VLA backbone to produce a comprehensive social interaction representation, which is then decoded by the flow-matching action expert, conditioned on the state encoder features and VLM embeddings, to generate velocity commands. The resulting command is executed through an RL-pretrained low-level locomotion controller.

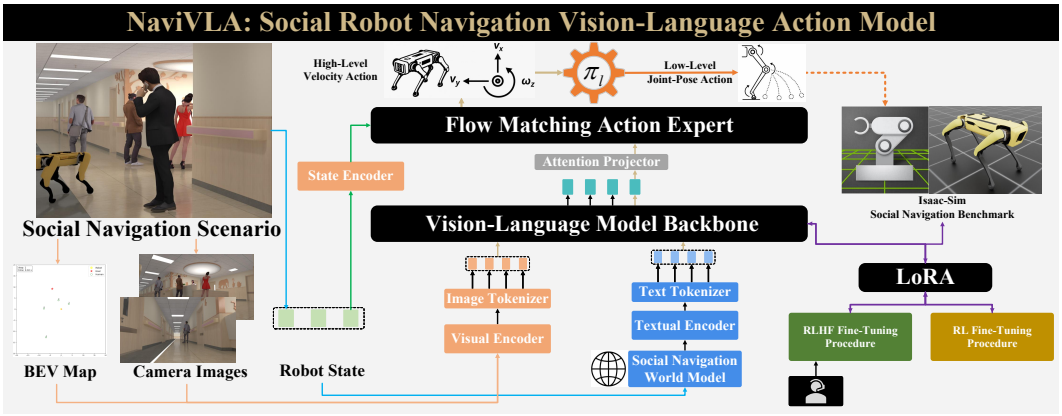



Figure 3: NaviVLA framework: NaviVLA is a social robot navigation VLA model that integrates multimodal observations and encodes them into social navigation action token features $H_t$ through a VLM backbone. The learned representation is then decoded by a flow matching action expert to generate high-level velocity actions $a_t^r$. Eventually, $a_t^r$ is converted into low-level joint pose commands $u_t^r$ by a motion planner $\pi_l$ to directly control robot motion. The entire framework is fine-tuned using RL and RLHF for socially compliant navigation.

**(b). Reinforcement Learning Fine-tuning Procedure:** Herein, we adopt Qwen3.5-9B Team (2026) as the VLM backbone. Nevertheless, our framework is model agnostic and can be readily extended to other VLM backbones Meta AI (2025); Abouelenin et al. (2025). Unlike recent VLA navigation methods Cheng et al. (2025); Zhang et al. (2025; 2024), which prompt the VLM to directly decode navigation actions in language form. NaviVLA does not rely on textual action generation, since such a prompt-based output interface is less suitable for end-to-end RL fine-tuning. Instead, NaviVLA extracts the hidden representations from VLM and learns the final robot control policy through RL, following the spirit of RL fine-tuned VLA policies such as $\pi^*_{0.6}$ Intelligence et al. (2025) and RLinf Zang et al. (2026). This design allows the model to preserve the semantic reasoning capability of the VLA backbone while optimizing continuous navigation actions under rewards.

Given the multimodal observation $o_t^r$, NaviVLA extracts token-level VLM hidden feature representations $H_t = f_{\text{VLM}}(o_t^r; \theta_{\text{VLM}}^{\text{freeze}}, \theta_{\text{LoRA}})$, where $H_t \in \mathbb{R}^{B\times L\times d_q}$ denotes a sequence of $L$ hidden feature tokens. To align feature dimensions, we leverage an attention pooling layer that compresses the full VLA feature sequence from $H_t$ to $\hat{H}_t \in \mathbb{R}^{B\times d_h}$. Specifically, a learnable navigation query attends to the all hidden tokens, assigns adaptive importance weights to each one, and incorporates them into the final representation $\hat{H}_t$, as follows:

$$\hat{H}_t = \sum_{i=1}^{L} \text{softmax}\left(\frac{q^\top W_k H_{t,i}}{\sqrt{d_k}}\right) W_v H_{t,i}, \qquad \hat{H}_t \in \mathbb{R}^{B\times d_h}. \tag{1}$$

where $q$ is a learnable attention query, and $W_k$ and $W_v$ are trainable projection matrices. The embedding $\hat{H}_t$ is then fed into flow matching action expert to parameterize a stochastic high-level robot policy, $\pi_h(a_t^r \mid H_t) = \mathcal{N}\left(a_t^r; \mu_h(\theta)(\hat{H}_t), \text{diag}(\sigma_h^2(\theta))\right)$, where $\sigma_h(\theta), \mu_h(\theta)$ denote the standard deviation and mean of robot policy $\pi_h$, separately.

The NaviVLA high-level policy $\pi_h(\theta)$ is optimized by maximizing the expected return $\mathcal{J}(\pi_h)$. We use PPO to update the policy with the clipped surrogate objective and the advantage $\hat{A}_t$ as follows:

$$\mathcal{L}_{\text{PPO}} = \mathbb{E}_t\left[\min\left(\frac{\pi_h(\theta)(a_t^r \mid H_t)}{\pi_h(\theta_{\text{old}})(a_t^r \mid H_t)}\hat{A}_t,\ \text{clip}\left(\frac{\pi_h(\theta)(a_t^r \mid H_t)}{\pi_h(\theta_{\text{old}})(a_t^r \mid H_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right]. \tag{2}$$

Eventually, PPO gradients are back propagated through the action log-probability and value losses to update VLM with the LoRA adapters Hu et al. (2022), the attention pooling layer, and the action expert.

**(c). Flow Matching Action Expert:** NaviVLA leverages a flow matching action expert to decode the VLA feature $\hat{H}_t$ with the stat encoder feature $\hat{H}_s$ into a continuous high-level velocity command $a_t^r$. The feature $\hat{H}_t$ is first projected into a flow condition $c_t = f_{mlp}(\hat{H}_t, \hat{H}_s)$. Conditioned on $c_t$,

the action expert parameterizes the vector field network $\mathcal{V}_\chi$ over the robot velocity action space $\mathcal{A}$. Starting from a deterministic initial action $x_{\text{init}} \in \mathbb{R}^3$, the conditional action flow with the scale $\lambda$ is formulated as follows:

$$\frac{dx_\tau}{d\tau} = \lambda \cdot \mathcal{V}_\chi \left( x_\tau, c_t, e(\tau) \right), \qquad x_1 = x_{\text{init}}, \qquad x_\tau \in \mathbb{R}^3, \tag{3}$$

where $x_\tau$ denotes the intermediate action state, $e(\tau)$ is the sinusoidal flow time embedding. The flow update procedure is defined as follows:

$$x_{\tau_{k+1}} = x_{\tau_k} - \lambda \cdot \Delta\tau \, \mathcal{V}_\chi \left( x_{\tau_k}, c_t, e(\tau_k) \right), \qquad \tau_k = 1 - k\Delta\tau, \qquad \Delta\tau = \frac{1}{K}. \tag{4}$$

After $K$ Euler reverse integration steps from $\tau = 1$ to $\tau = 0$, the final refined action state is used as the mean of the high-level stochastic policy $\pi_h(a_t^r \mid H_t) = \mathcal{N}\left(a_t^r; \mu_h(\theta)(\hat{H}_t), \text{diag}\left(\sigma_h^2(\theta)\right)\right) = \mathcal{N}\left(a_t^r; x_{\tau_K}, \text{diag}\left(\sigma_h^2(\theta)\right)\right)$. To remain compatible with PPO, the flow expert itself outputs a deterministic action mean of distribution, while the stochasticity required for exploration and policy ratio computation is introduced by the external Gaussian policy.

**(d). Reinforcement Learning from Human Feedback Procedure:** Desipte PPO optimizes the navigation policy with hand-crafted rewards which are insufficient to fully encode underlying social norms. To further align NaviVLA with pedestrians' preferences, we introduce an online DPO-based (direct preference optimization) RLHF fine-tuning stage. In this stage, we freeze the VLM backbone and update the LoRA adapters together with the other components, to encode social norms.

During online RLHF fine-tuning process, NaviVLA samples multiple trajectory rollouts under identical navigation scenario. For two candidate trajectories $\xi_i$ and $\xi_j$, we automatically assign a preference label according to evaluation metrics, such as uncomfortable rate, social distance violation, and social score. In particular, if trajectory $\xi_i$ has a lower uncomfortable rate, fewer social violations, and higher social score than $\xi_j$, then $\xi_i$ is treated as the preferred trajectory $\xi^+$ and $\xi_j$ as the rejected trajectory $\xi^-$. This produces an online preference dataset $\mathcal{D}_{\text{pref}} = \{(\xi_1^+, \xi_1^-), \cdots, (\xi_B^+, \xi_B^-)\}$ with batch $B$, in which each preference label can be automatically assigned according to social norm evaluation metrics, or directly provided by human supervisors.

Given a trajectory $\xi = (H_0, a_0^r, \cdots, H_T, a_T^r)$, its policy log-likelihood is computed as $\log \pi_h(\xi) = \sum_{t=0}^{T} \log \pi_h(a_t^r \mid H_t)$. We then optimize the LoRA-adapted NaviVLA policy as follows:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(\xi^+, \xi^-)} \left[ \log \sigma \left( \beta \left[ \log \frac{\pi_h(\xi^+; \theta_{VLM}^{freeze}, \theta_{\text{LoRA}})}{\pi_{\text{ref}}(\xi^+)} - \log \frac{\pi_h(\xi^-; \theta_{VLM}^{freeze}, \theta_{\text{LoRA}})}{\pi_{\text{ref}}(\xi^-)} \right] \right) \right], \tag{5}$$

where $\pi_{\text{ref}}$ is the frozen reference policy before alignment, $\beta$ controls the strength of optimization, and $\theta_{\text{LoRA}}$ denotes the trainable LoRA parameters. This objective increases the relative likelihood of socially preferred trajectories while suppressing trajectories that cause uncomfortable interactions or violate social norms. Therefore, the online DPO stage serves as a social norm alignment module for NaviVLA, complementing PPO-based reward optimization with preference feedback.

## 4 Experiments

**(a). Setup:** We conduct simulation experiments to evaluate the NaviVLA planner in the Isaac-Sim NVIDIA (2026) benchmark, covering diverse crowd scenarios. In each episode, the robot is initialized with a start pose and a target goal, while pedestrians are spawned with different initial positions, motion policies, and destinations. The robot is required to reach the goal efficiently while avoiding collisions, maintaining comfortable distances from humans, and satisfying social navigation constraints. We compare NaviVLA with a set of existing SOTA path planner baselines. For fair comparison, all planners are equipped with the same pretrained low-level locomotion controller and PPO trainer.

**(b). Baselines:** For traditional planners, A* Hart et al. (1968) and RRT* Karaman & Frazzoli (2011) are originally designed for static environments. Therefore, we adapt them to social navigation using receding horizon replanning. At each replanning step, the planner updates the local map using the latest observations, inflates pedestrians as static obstacles, replans a clear path, and tracks a local waypoint to generate velocity commands. However, these methods remain reactive, which they do




Figure 4: Physical robot configuration and real-world demonstrations.

not reason about social interactions, which may lead to conservative, oscillatory, or inefficient behaviors in crowded scenes. In contrast, Social Force Helbing & Molnar (1995) and ORCA Alonso-Mora et al. (2013) are conventional physics-based or geometry-based dynamic navigation baselines, and can be naturally integrated into our benchmark and experiments.

For learning-based frameworks, the MLP Rumelhart et al. (1986) and Transformer Vaswani et al. (2017) policies without explicit HRI encoders are utilized to highlight the importance of specific HRI modeling in policy network design. CADRL Chen et al. (2017) learns latent HRI representations through individual pairwise interaction encoder blocks. LSTM-RL Everett et al. (2018) and SARL Chen et al. (2019) further incorporate recurrent networks and attention mechanisms, respectively, to model interactions between the robot and the entire crowd. Moreover, DSRNN Liu et al. (2021) and NaviSTAR Wang et al. (2023) leverage the spatio-temporal graph to structural construct HRI representation via the social attention network Vemula et al. (2018) or spatial-temporal transformer network. More currently, NaviDIFF Wang et al. (2025b) is proposed to capture the physical-HRI features, using a port-Hamiltonian mechanics constrained diffusion network Mao et al. (2023).

**(c). Evaluation Metrics:** According to Section. 3.1, total $N$ times independent experiments are mainly evaluated by following metrics: success rate $SR = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(i = success)$; social score $SC$; uncomfortable rate $UR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{t=1}^{T_i}\mathbf{1}[\min_h d_{i,h}^t < d_{\text{social}}]$; and navigation time ,$NT = \frac{1}{N}\sum_{i=1}^{N} T_i$. The detailed definition of $SC$ is provided in Appendix.

**(d). Experiment Procedure:** The *Human-Diffusion* benchmark contains several dataset-driven crowd policies, including ETH, HOTEL, UNIV, ZARA01, ZARA02, GCS, SDD, and EDIN. The rule-based human simulator includes eight scene-policy combinations: *Circle*+SF, *Random*+SF, *Corridor*+SF, *Group*+SF, *Circle*+ORCA, *Random*+ORCA, *Corridor*+ORCA, and *Group*+ORCA. All trainable planners are trained only on random+ORCA and the ETH *Human-Diffusion* scenario, and are evaluated on all remaining scenarios to test generalization.

All VLA-based models are trained on eight H100 GPUs. We train NaviVLA using a frozen-to-LoRA strategy, where the model is first optimized with the VLM backbone frozen, and once the performance stabilizes, LoRA adaptation is enabled to jointly fine-tune the VLM with the entire framework.

## 5 Results

Table 1 reports the quantitative results over both *Human-Diffusion* and rule-based crowd simulator scenarios. Overall, NaviVLA achieves the highest social compliance performance, with an average $SC$ of 75.36, outperforming all baselines. It also improves over NaviVLA without DPO by 8.08 points, demonstrating the effectiveness of the RLHF-based social norm encoder for aligning robot behaviors with human social preferences.

For navigation success, NaviVLA without DPO obtains the highest average success rate of 0.810, while the full NaviVLA achieves a slightly lower but still competitive success rate of 0.790. This

Table 1: Total 1000 individual identical experiment table of NaviVLA with other baselines over *Human-Diffusion* and crowd simulator scenarios. Where STF is spatial-temporal feature encoder, and ◐, ◑ denote the spatial encoder and Temporal encoder.

| | Key Attributes | | | | Human Diffusion AVG | | | | Crowd Simulator AVG | | | | AVG | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | VLA | BEV | STF | RLHF | SR | SC | NL | NT | SR | SC | NL | NT | SR | SC | NL | NT |
| A* (Hart et al., 1968) | × | ⊗ | ○ | ⊗ | 0.315 | 54.16 | 11.94 | 12.10 | 0.135 | 39.84 | 9.70 | 10.12 | 0.225 | 47.00 | 10.82 | 11.11 |
| RRT* (Karaman & Frazzoli, 2011) | × | ⊗ | ○ | ⊗ | 0.309 | 53.70 | 12.23 | 12.50 | 0.318 | 55.14 | 6.22 | 6.57 | 0.314 | 54.42 | 9.22 | 9.54 |
| Social Force (Helbing & Molnar, 1995) | × | ⊗ | ○ | ⊗ | 0.314 | 53.93 | 11.94 | 12.25 | 0.298 | 54.16 | 5.80 | 6.15 | 0.306 | 54.05 | 8.87 | 9.20 |
| ORCA (Alonso-Mora et al., 2013) | × | ⊗ | ○ | ⊗ | 0.439 | 60.37 | 14.62 | 15.64 | 0.349 | 53.97 | 22.40 | 29.00 | 0.394 | 57.17 | 18.51 | 22.32 |
| MLP (Rumelhart et al., 1986) | × | ⊗ | ◐ | ⊗ | 0.04 | 5.16 | 16.31 | 15.73 | 0.04 | 4.97 | 10.15 | 9.23 | 0.04 | 5.07 | 13.23 | 12.48 |
| Transformer (Vaswani et al., 2017) | × | ⊗ | ◐ | ⊗ | 0.12 | 10.85 | 13.56 | 12.15 | 0.06 | 7.73 | 11.37 | 11.08 | 0.09 | 9.29 | 12.47 | 11.62 |
| CADRL (Chen et al., 2017) | × | ⊗ | ◐ | ⊗ | 0.547 | 32.22 | 35.06 | 23.81 | 0.361 | 13.18 | 24.54 | 14.82 | 0.454 | 22.70 | 29.80 | 19.31 |
| LSTM-RL (Everett et al., 2018) | × | ⊗ | ● | ⊗ | 0.297 | 14.10 | 39.08 | 27.09 | 0.401 | 18.24 | 27.90 | 17.12 | 0.349 | 16.17 | 33.49 | 22.11 |
| SARL (Chen et al., 2019) | × | ⊗ | ◐ | ⊗ | 0.133 | 7.38 | 67.64 | 38.32 | 0.351 | 16.10 | 37.60 | 23.31 | 0.242 | 11.74 | 52.62 | 30.81 |
| DSRNN (Liu et al., 2021) | × | ⊗ | ● | ⊗ | 0.484 | 26.29 | 33.12 | 23.73 | 0.289 | 9.39 | 23.44 | 14.40 | 0.386 | 17.84 | 28.28 | 19.07 |
| NaviSTAR (Wang et al., 2023) | × | ⊗ | ● | ● | 0.692 | 57.41 | 33.79 | 27.98 | 0.690 | 48.09 | 31.01 | 24.00 | 0.691 | 52.75 | 32.40 | 25.99 |
| NaviDIFF (Wang et al., 2025b) | × | ⊗ | ● | ● | 0.655 | 53.27 | 30.53 | 22.78 | 0.710 | 50.12 | 36.29 | 27.61 | 0.683 | 51.70 | 33.41 | 25.20 |
| NaviVLA (Qwen3.5-9B) wo DPO | ✓ | ● | ● | ⊗ | **0.925** | 87.21 | 23.96 | 26.35 | **0.694** | 47.35 | 22.88 | 25.86 | **0.809** | 67.28 | 23.42 | 26.11 |
| NaviVLA (Qwen3.5-9B) | ✓ | ● | ● | ● | 0.906 | **93.80** | 24.71 | 27.18 | 0.674 | **56.91** | 23.52 | 26.69 | 0.790 | **75.36** | 24.12 | 26.94 |

indicates that the DPO-based social norm encoder makes the robot more conservative and socially compliant, leading to improved social compliance at the cost of slightly reduced goal-reaching efficiency. This trend is also reflected in the navigation length and navigation time, where NaviVLA produces longer trajectories and larger $NT$ values than NaviVLA without DPO. Specifically, NaviVLA increases the average $NL$ from $23.42$ to $24.12$ and the average $NT$ from $26.11$ to $26.94$.

Compared with classical planners such as A*, RRT*, Social Force, and ORCA, learning-based methods show different levels of adaptability in complex dynamic scenarios. However, simple MLP and Transformer policies still perform poorly, showing that directly learning navigation from limited spatial features is insufficient for complex social navigation. Among prior RL-based methods, NaviSTAR achieves a competitive average success rate of $0.691$, and NaviDIFF also maintains a strong success rate of $0.683$. However, their social compliance scores remain substantially lower than NaviVLA, with average $SC$ values of $52.75$ and $51.70$, respectively. This suggests that conventional RL reward design alone may be insufficient to capture implicit social norms. By integrating VLA reasoning, BEV-based spatial context, spatial-temporal feature encoding, and RLHF-based preference alignment, NaviVLA provides a stronger balance between goal reaching and social compliance.

The results also show consistent generalization across both benchmark types. In *Human-Diffusion* scenarios, NaviVLA achieves the highest $SC$ of $93.80$, while in rule-based crowd simulator scenarios, it also obtains the highest $SC$ of $56.91$. Since all trainable planners are trained only on Random+ORCA and ETH *Human-Diffusion* and evaluated on unseen scenarios, these results demonstrate that NaviVLA can generalize to different crowd dynamics. Eventually, the physical robot configuration and a simple test are shown in Fig. 4.

## 6 Conclusion

In this paper, we proposed **NaviVLA**, an RL-trained VLA-based planner for socially-aware robot navigation. NaviVLA integrates multimodal observation with a flow matching action expert to generate continuous robot velocity commands. We further introduced an RLHF-based social norm encoder that guides the policy toward human-preferred navigation behaviors. In addition, an IsaacSim-based social navigation benchmark is developed to enable scalable and realistic evaluation across diverse crowd dynamics. Experimental results in simulation and on physical robots demonstrate that NaviVLA improves navigation safety, and social compliance compared with existing baselines. These results show the potential of VLA-based planners as a unified framework for socially compliant embodied navigation in human-populated environments.

## REFERENCES


Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul Beardsley, and Roland Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Distributed autonomous robotic systems: The 10th international symposium*, pp. 203–216. Springer, 2013.

Karl Johan Åström. Optimal control of markov processes with incomplete state information i. *Journal of mathematical analysis and applications*, 10:174–205, 1965.

Inhwan Bae, Junoh Lee, and Hae-Gon Jeon. Continuous locomotive crowd behavior generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 22416–22431, 2025.

Bonny Banerjee and B Chandrasekaran. A framework of voronoi diagram for planning multiple paths in free space. *Journal of Experimental & Theoretical Artificial Intelligence*, 25(4):457–475, 2013.

Abhijat Biswas, Allan Wang, Gustavo Silvera, Aaron Steinfeld, and Henny Admoni. Socnavbench: A grounded simulation testing framework for evaluating social navigation. *ACM Transactions on Human-Robot Interaction (THRI)*, 11(3):1–24, 2022.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi_0$: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Silvère Bonnabel, Philippe Martin, and Erwan Salaün. Invariant extended kalman filter: Theory and application to a velocity-aided attitude estimation problem. In *Proceedings of the 48th IEEE Conference on Decision and Control*, pp. 1297–1304. IEEE, 2009. doi: 10.1109/CDC.2009.5400372.

Boston Dynamics. Spot: The agile mobile robot. https://bostondynamics.com/products/spot/.

Chao Cao, Peter Trautman, and Soshi Iba. Dynamic channel: A planning framework for crowd navigation. In *2019 international conference on robotics and automation (ICRA)*, pp. 5551–5557. IEEE, 2019.

Yuhang Che, Allison M Okamura, and Dorsa Sadigh. Efficient and trustworthy social navigation via explicit and implicit robot–human communication. *IEEE Transactions on Robotics*, 36(3): 692–707, 2020.

Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6015–6022, 2019. doi: 10.1109/ICRA.2019.8794134.

Changan Chen, Sha Hu, Payam Nikdel, Greg Mori, and Manolis Savva. Relational graph learning for crowd navigation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10007–10013. IEEE, 2020.

Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P. How. Socially aware motion planning with deep reinforcement learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1343–1350, 2017. doi: 10.1109/IROS.2017.8202312.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Xueyan Zou, Jan Kautz, Erdem Biyik, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

Gregory F Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2-3):393–405, 1990.

Matej Dobrevski and Danijel Skočaj. Dynamic adaptive dynamic window approach. *IEEE Transactions on Robotics*, 40:3068–3081, 2024.

Noel E Du Toit and Joel W Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 28(1):101–115, 2011.

Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3052–3059. IEEE, 2018.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51 (5):4282, 1995.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 38–44. IEEE, 2016.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. Pi0.6: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

Jeffrey Kane Johnson. The colliding reciprocal dance problem: A mitigation strategy with application to automotive active safety systems. In *2020 American Control Conference (ACC)*, pp. 1417–1422. IEEE, 2020.

Marcelo Kallmann. Path planning in triangulations. In *Proceedings of the IJCAI workshop on reasoning, representation, and learning in computer games*, pp. 49–54, 2005.

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

Kapil D Katyal, Gregory D Hager, and Chien-Ming Huang. Intent-aware pedestrian prediction for adaptive crowd navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3277–3283. IEEE, 2020.

Lydia E Kavraki, Mihail N Kolountzakis, and J-C Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and automation*, 14(1):166–171, 1998.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016a.

Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016b.

Markus Kuderer, Henrik Kretzschmar, Christoph Sprunk, and Wolfram Burgard. Feature-based prediction of trajectories for socially compliant navigation. *Robotics: Science and Systems VIII*, 8, 2012.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *International Conference on Machine Learning*, 2021a.

Kimin Lee, Laura Smith, Anca Dragan, and Pieter Abbeel. B-pref: Benchmarking preference-based reinforcement learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021b. URL https://openreview.net/forum?id=ps95-mkHF_.

Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer graphics forum*, volume 26, pp. 655–664. Wiley Online Library, 2007.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Shuijing Liu, Peixin Chang, Weihang Liang, Neeloy Chakraborty, and Katherine Driggs-Campbell. Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3517–3524, 2021. doi: 10.1109/ICRA48506.2021.9561595.

Shangyi Luo, Peng Sun, Ji Zhu, Yuhong Deng, Cunjun Yu, Anxing Xiao, and Xueqian Wang. Gson: a group-based social navigation framework with large multimodal model. *IEEE Robotics and Automation Letters*, 2025.

Barbara Majecka. *Statistical models of pedestrian behaviour in the forum*. PhD thesis, Master's thesis, School of Informatics, University of Edinburgh, 2009.

Weibo Mao, Chenxin Xu, Qi Zhu, Siheng Chen, and Yanfeng Wang. Leapfrog diffusion model for stochastic trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5517–5526, 2023.

Christoforos Mavrogiannis, Krishna Balasubramanian, Sriyash Poddar, Anush Gandra, and Siddhartha S Srinivasa. Winding through: Crowd navigation via topological invariance. *IEEE Robotics and Automation Letters*, 8(1):121–128, 2022.

Christoforos I Mavrogiannis and Ross A Knepper. Multi-agent path topology in support of socially competent navigation planning. *The International Journal of Robotics Research*, 38(2-3):338–356, 2019.

Meta AI. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 2025.

Max Muchen Sun, Francesca Baldini, Katie Hughes, Peter Trautman, and Todd Murphey. Mixed strategy nash equilibrium for crowd navigation. *The International Journal of Robotics Research*, 44(7):1156–1185, 2025.

Michael J. Munje, Chen Tang, Shuijing Liu, Zichao Hu, Yifeng Zhu, Jiaxun Cui, Garrett Warnell, Joydeep Biswas, and Peter Stone. Socialnav-sub: Benchmarking vlms for scene understanding in social robot navigation. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2025. Project page: https://larg.github.io/socialnav-sub.

Siddarth Narasimhan, Aaron Hao Tan, Daniel Choi, and Goldie Nejat. Olivia-nav: An online lifelong vision language approach for mobile robot social navigation. In *2025 IEEE international conference on robotics and automation (ICRA)*, pp. 9130–9137. IEEE, 2025.

NVIDIA. NVIDIA Isaac Sim: Robotics simulation and synthetic data generation. https://developer.nvidia.com/isaac/sim, 2026. Accessed: 2026-05-24.

Andreas Opedal, Niklas Stoehr, Abulhair Saparov, and Mrinmaya Sachan. World models for math story problems. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 9088–9115, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.579. URL https://aclanthology.org/2023.findings-acl.579.

Amirreza Payandeh, Daeun Song, Mohammad Nazeri, Jing Liang, Praneel Mukherjee, Amir Hossain Raj, Yangzhe Kong, Dinesh Manocha, and Xuesu Xiao. Social-llava: Enhancing social robot navigation through human-language reasoning. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 17192–17198. IEEE, 2025.

Noé Pérez-Higueras, Roberto Otero, Fernando Caballero, and Luis Merino. Hunavsim: A ros 2 human navigation simulator for benchmarking human-aware robot navigation. *IEEE robotics and automation letters*, 8(11):7130–7137, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Jorge Rios-Martinez, Anne Spalanzani, and Christian Laugier. From proxemics theory to socially-aware navigation: A survey. *International Journal of Social Robotics*, 7(2):137–153, 2015.

Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pp. 549–565. Springer, 2016.

Christoph Rösmann, Wendelin Feiten, Thomas Wösch, Frank Hoffmann, and Torsten Bertram. Efficient trajectory optimization using a sparse model. In *2013 European conference on mobile robots*, pp. 138–143. IEEE, 2013.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Sepehr Samavi, James R Han, Florian Shkurti, and Angela P Schoellig. Sicnav: Safe and interactive crowd navigation using model predictive control and bilevel optimization. *IEEE Transactions on Robotics*, 41:801–818, 2024.

Abulhair Saparov and Tom M Mitchell. Towards general natural language understanding with probabilistic worldbuilding. *Transactions of the Association for Computational Linguistics*, 10:325–342, 2022.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9339–9347, 2019.

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

Volodymyr Shcherbyna, Linh Kastner, Diego Diaz, Huu Giang Nguyen, Maximilian Ho-Kyoung Schreff, Tim Seeger, Jonas Kreutz, Ahmed Martban, Zhengcheng Shen, Huajian Zeng, et al. Arena 4.0: A comprehensive ros2 development and benchmarking platform for human-centric navigation using generative-model-based environment generation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9138–9144. IEEE, 2025.

Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, et al. igibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7520–7527. IEEE, 2021.

Daeun Song, Jing Liang, Amirreza Payandeh, Amir Hossain Raj, Xuesu Xiao, and Dinesh Manocha. Vlm-social-nav: Socially aware robot navigation through scoring using vision-language models. *IEEE Robotics and Automation Letters*, 10(1):508–515, 2024.

Muchen Sun, Francesca Baldini, Peter Trautman, and Todd Murphey. Move Beyond Trajectories: Distribution Space Coupling for Crowd Navigation. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.053.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Pete Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015.

Annemarie Turnwald and Dirk Wollherr. Human-like motion planning based on game theoretic decision making. *International Journal of Social Robotics*, 11:151–170, 2019.

Unitree Robotics. Unitree go2 quadruped robot. `https://www.unitree.com/go2/`.

Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE international conference on robotics and automation*, pp. 1928–1935. Ieee, 2008.

Pieter van Goor, Tarek Hamel, and Robert Mahony. Equivariant filter (eqf). *IEEE Transactions on Automatic Control*, 68(6):3501–3512, 2023. doi: 10.1109/TAC.2022.3194094.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *2018 IEEE international Conference on Robotics and Automation (ICRA)*, pp. 4601–4607. IEEE, 2018.

Allan Wang, Christoforos Mavrogiannis, and Aaron Steinfeld. Group-based motion prediction for navigation in crowded environments. In *Conference on Robot Learning*, pp. 871–882. PMLR, 2022.

Weizheng Wang, Ruiqi Wang, Le Mao, and Byung-Cheol Min. Navistar: Socially aware robot navigation with hybrid spatio-temporal graph transformer and preference learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11348–11355, 2023. doi: 10.1109/IROS55552.2023.10341395.

Weizheng Wang, Le Mao, Ruiqi Wang, and Byung-Cheol Min. Multi-robot cooperative socially-aware navigation using multi-agent reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12353–12360. IEEE, 2024.

Weizheng Wang, Aniket Bera, and Byung-Cheol Min. Hypergraph-based coordinated task allocation and socially-aware navigation for multi-robot systems. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16225–16232. IEEE, 2025a.

Weizheng Wang, Chao Yu, Yu Wang, and Byung-Cheol Min. Human-robot cooperative distribution coupling for hamiltonian-constrained social navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10808–10815. IEEE, 2025b.

Weizheng Wang, Obi Ike, Soyun Choi, Sungeun Hong, Aniket Bera, and Byung-Cheol Min. Logic-guided socially-aware robot navigation world model, 2026. URL https://arxiv.org/abs/2510.23509.

Congcong Wen, Yifan Liu, Geeta Chandra Raju Bethala, Shuaihang Yuan, Hao Huang, Yu Hao, Mengyu Wang, Yu-Shen Liu, Anthony Tzes, and Yi Fang. Socially-aware robot navigation enhanced by bidirectional natural language conversations using large language models. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 15314–15321. IEEE, 2025.

Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9068–9079, 2018.

Jiannan Xiang, Tianhua Tao, Yi Gu, Tianmin Shu, Zirui Wang, Zichao Yang, and Zhiting Hu. Language models meet world models: Embodied experiences enhance language models. *Advances in neural information processing systems*, 36, 2024.

Zhanteng Xie and Philip Dames. Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles. *IEEE Transactions on Robotics*, 39(4):2700–2719, 2023.

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, Zhihao Liu, Kang Chen, Wenhao Tang, Quanlu Zhang, Weinan Zhang, Chao Yu, and Yu Wang. Rlinf-vla: A unified and efficient framework for reinforcement learning of vision-language-action models. In *Robotics: Science and Systems*, 2026.

Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. *Robotics: Science and Systems*, 2024.

Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks. *Robotics: Science and Systems (RSS)*, 2025.

Bolei Zhou, Xiaogang Wang, and Xiaoou Tang. Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 2871–2878. IEEE, 2012.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. WALL-e: World alignment by neurosymbolic learning improves world model-based LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=DorAT49sxj.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot*

*Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

# SUPPLEMENTARY MATERIALS OF NAVIVLA

**A Problem Statement:** **17**

**B Social Navigation Benchmark Appendix** **18**

B.1 Benchmark Overview . . . . . 18

B.2 Robot Configuration . . . . . 19

B.3 Human Simulator . . . . . 24

B.4 Baselines . . . . . 31

B.5 Learning curve . . . . . 44

B.6 Evaluation Metrics . . . . . 46

B.7 PPO Configuration . . . . . 48

**C The Grid-Based Discrete Social Navigation Experiments** **50**

C.1 Task Configuration . . . . . 50

C.2 RL Fine-Tuning Procedure for the VLA Policy . . . . . 53

C.3 RL-VLA Fine-Tuning Procedure Experimental nalysis . . . . . 54

C.4 RLHF Fine-Tuning Procedure for the VLA Policy . . . . . 55

**D NaviVLA Appendix** **58**

D.1 Robot Input . . . . . 58

D.2 NaviVLA RL Fine-Tuning Framework . . . . . 65

**E Experiments** **67**

**F Acknowledge** **70**

# A PROBLEM STATEMENT:

Robot navigation is the task that aims to search a set of collision-free and efficient routes from an initial location to the destination. With the developments of robotics and AI technologies, deploying social robots in human-filled public spaces, while tailoring robot behaviors with respect to human expectations and preferences has become a fundamental problem for advanced robotic operation and embodied intelligence, known as socially aware robot navigation task. Herein, we formulate it as navigation in a dynamic human environment, where each agent has both private intent, such as behavioral strategy, preference, and goal, and public information, such as current position and velocity. The robot is required to reach its destination without colliding with pedestrians while following social norms, pedestrian preferences, and implicit expectations. Thus, we model the problem as a non-communicative multi-agent system for cooperative collision avoidance, which can be addressed as a Dec-POMDP under the RL paradigm. Based on this formulation, we propose NaviVLA, a coupled HRI inference and path-planning framework.

Let $\mathcal{S} \subseteq \mathbb{R}^2$ presents the available spaces of the environment. The social robot who occupies an physical area $\mathcal{S}_r$ has to search a set of clear paths $\tau \in \mathcal{T}$ within a group of pedestrians $\mathcal{S}_1, \cdots, \mathcal{S}_N$ from the origin $s_o \in \mathcal{S}_0$ to goal $s_g \in \mathcal{S}$. Eventually, the objective $\mathcal{J}$ of the socially-aware navigation task is defined as follows:

$$\begin{aligned} \mathcal{J} &= \underset{\forall\{\tau_i \in \mathcal{T}\}}{\arg\min}\ c^t(\tau_i) + c^s(\tau_i) \\ &= \underset{\pi_h(\theta)}{\arg\max}\, \mathbb{E}[\textstyle\sum_t \gamma^t \mathcal{R}(s_t^r, a_t^r)] \\ &\text{s.t. } \forall\, h \in [human_1, human_N] \\ &dis_{r,h} > \rho_r + \rho_h;\ \ dis_{r,g} < \rho_r;\ \ dis_{r,s} < \rho_r + \rho_s \end{aligned} \tag{6}$$

where the clear path $\tau$ is sampled from the navigation planner's clear path region $\mathcal{T}$. The navigation time cost function and social compliance cost function are abstracted as $c^t(\cdot)$ and $c^s(\cdot)$, respectively. Subsequently, the optimization task of the social robot's cost functions is converted into the social robot navigation expectation function within the Dec-POMDP framework. The feasibly navigating conditions are defined as collision avoidance between the robot and pedestrians or static obstacles, where the relative distance is lower than the robot radius and human radius, $dis_{r,h} > \rho_r + \rho_h$) or lower than the robot radius and static obstacles radius, $dis_{r,s} < \rho_r + \rho_s$). Additionally, $dis_{i,g} < \rho_i$ indicates that the social robot has reached the destination. Formally, the navigation inference approaches of the robot system are relaxed as a Dec-POMDP optimization problem, in which the action execution policy $\pi_h$ of the social robot is trained to maximize the expected reward function $\mathcal{R}$ and minimize the cost function value.

# B Social Navigation Benchmark Appendix

## B.1 Benchmark Overview

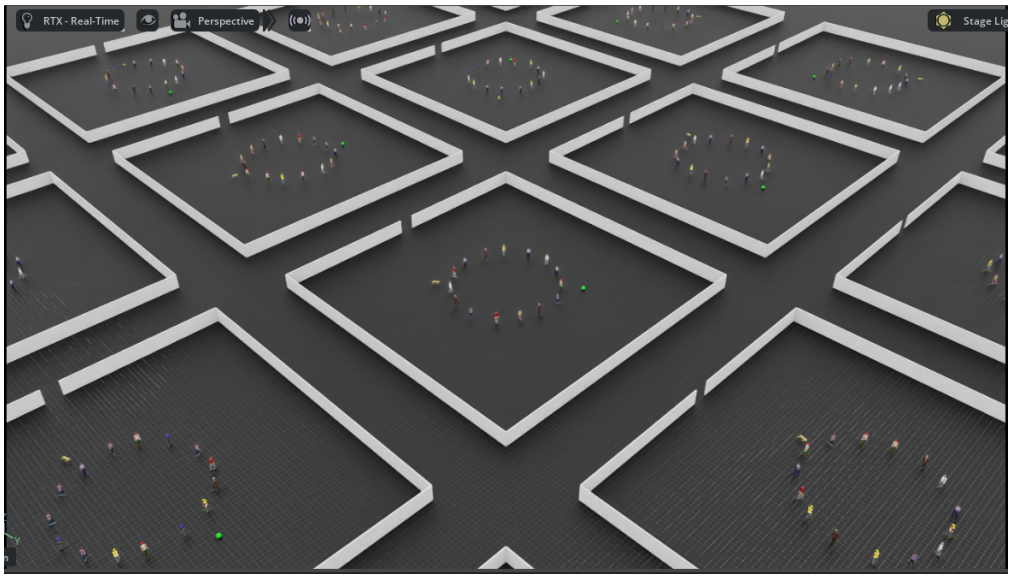



Figure 5: Visualization of the IsaacSim-based social navigation benchmark with parallel environments.

As shown in Fig. 5, we develop an IsaacSim-based social navigation benchmark to evaluate robot navigation policies in dynamic human-shared environments. The benchmark provides multiple human dynamics, including crowd circle crossing, random crowd motion, bi-directional crowd flow (corridor), multi-group scenarios, and Human Diffusion. Each configuration contains a quadruped robot, a navigation target, and multiple pedestrians.

The robot is controlled through a hierarchical control structure. The high-level planner outputs continuous velocity commands $[v_x, v_y, \omega_z]$, where $v_x$ and $v_y$ are planar linear velocities and $\omega_z$ is the yaw angular velocity. These commands are then executed by a pretrained low-level locomotion controller for the Spot/Anymal/Go2 quadruped robot.

Pedestrian motion is generated by different human simulators, including ORCA, Social Force, and Human Diffusion trajectory. ORCA and Social Force provide rule-based crowd behaviors, while Human Diffusion introduces data-driven pedestrian trajectories for more realistic crowd motion.

The observation space of VLA-based planners includes BEV maps, multi-view RGB-D camera observations, and structured world-model text descriptions that is converted from robot observation, involving robot state, goal position, nearby human positions and velocities information. Moreover, the robot observation can be directly scanned by other traditional or learning-based planners.

The benchmark allows fair comparison among traditional planners, learning-based methods, and proposed NaviVLA policy, including A*, RRT*, ORCA, Social Force, CADRL, LSTM-RL, SARL, DSRNN, NaviSTAR, NaviDIFF, and NaviVLA. Benchmark evaluates each method using success rate, collision rate, uncomfortable interaction rate, navigation time, path length, minimum human distance, and a unified social navigation score.

### B.2 Robot Configuration

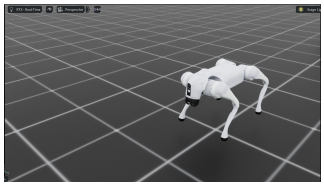
#### B.2.1 Sensor Robot information



(a) Spot

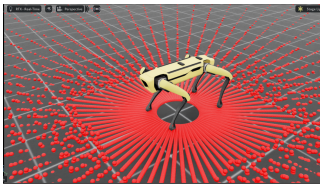


(b) Anymal

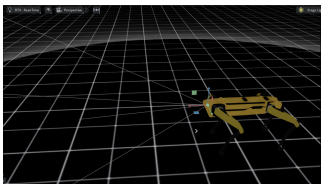


(c) Go2

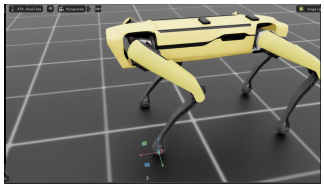
Figure 6: Visualization of the quadruped robot platforms used in the IsaacSim social navigation benchmark, including Spot, ANYmal, and Go2.



(a) RayCaster Sensor



(b) Camera Sensor



(c) Contact Sensor

Figure 7: Visualization of the onboard sensors used in the IsaacSim social navigation benchmark, including RayCaster sensor, RGB-D cameras, and contact sensors.

As shown in Fig. 6, the benchmark offers three quadruped robot platforms for social navigation scenarios, who are Boston Dynamic Spot Boston Dynamics, Anymal Hutter et al. (2016), and Unitree Go2 Unitree Robotics.

For contact perception, the robot is equipped with contact sensors attached to its rigid bodies. For the Spot configuration, the body link is used as the robot base, the foot links, and the upper/lower leg links. This allows the system to detect robot body contact, foot contact, and abnormal leg collisions.

For visual perception, four RGB-D cameras are mounted on the Spot body, facing the front, left, rear, and right directions. These cameras provide approximately 360-degree surrounding perception. Each camera outputs RGB images and depth images, and the four views can be stacked as multi-view robot observations. The camera configuration uses a horizontal field of view of about 95 degrees and a visible depth range from 0.1 m to 5.0 m.

The robot is also equipped with a RayCaster-based perception sensor, which works similarly to a lightweight LiDAR by casting rays from the robot body and detecting intersections with surrounding objects. By changing the ray pattern and target collision meshes, this sensor can be used not only for terrain perception but also for estimating distances to nearby humans and obstacles which can be utilized to generate the robot BEV perception maps, providing useful proximity information for socially aware navigation.

The robot is controlled using a hierarchical command structure. The high-level planner $\pi_h$ generates velocity commands for the low-level motion controller $\pi_l$, which executes the corresponding 12-dimensional joint actions for the robot legs. The input to the high-level planner includes the robot state $s_t^r = [p_x, p_y, g_x, g_y, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]$, the observed human states, and the scene layout information. The high-level action is defined as a velocity command: $a_t^r = [v_x, v_y, \omega_z]$. Moreover, for NaviVLA, the high-level planner additionally takes multi-sensor observations and the BEV map as input, enabling the policy to reason from both semantic and spatial scene representations.

#### B.2.2 Motion Planner Training Environment and Reward Functions

The low-level motion planner $\pi_l$ is trained in a flat-ground robot simulator using PPO. The task is formulated as a velocity tracking problem, where the policy receives a target body frame velocity

command $\mathbf{u}_t = \left[v_{x,t}^{\mathrm{cmd}}, v_{y,t}^{\mathrm{cmd}}, \omega_{z,t}^{\mathrm{cmd}}\right]$, and outputs low-level robot actions to drive the quadruped robot. Here, $v_{x,t}^{\mathrm{cmd}}$ and $v_{y,t}^{\mathrm{cmd}}$ denote the desired planar linear velocity, and $\omega_{z,t}^{\mathrm{cmd}}$ denotes the desired yaw rate. The goal of the motion planner is to make the actual robot velocity $\mathbf{v}_{xy,t}$ and yaw rate $\omega_{z,t}$ accurately track the commanded velocity while maintaining stable and smooth locomotion.

The main tracking reward consists of a planar velocity tracking term and a yaw-rate tracking term:

$$r_{\mathrm{lin},t} = \exp\left(-\frac{\left\|\mathbf{v}_{xy,t}^{\mathrm{cmd}} - \mathbf{v}_{xy,t}\right\|_2^2}{\sigma_{\mathrm{lin}}}\right), \quad r_{\mathrm{yaw},t} = \exp\left(-\frac{\left(\omega_{z,t}^{\mathrm{cmd}} - \omega_{z,t}\right)^2}{\sigma_{\mathrm{yaw}}}\right), \tag{7}$$

where $\sigma_{\mathrm{lin}}$ and $\sigma_{\mathrm{yaw}}$ are tracking temperature parameters.

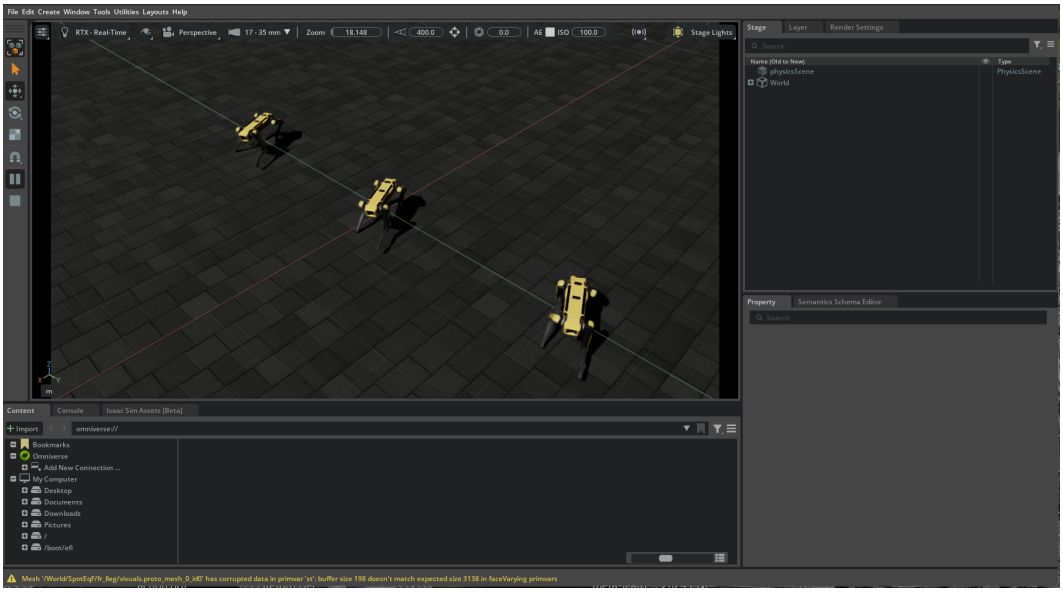
To improve locomotion stability and smoothness, several regularization terms are further introduced $r_{z,t} = \|v_{z,t}\|_2^2$, $r_{\omega xy,t} = \|\boldsymbol{\omega}_{xy,t}\|_2^2$, $r_{\tau,t} = \|\boldsymbol{\tau}_t\|_2^2$, $r_{\ddot{q},t} = \|\ddot{\mathbf{q}}_t\|_2^2$, $r_{\Delta a,t} = \|\mathbf{a}_t - \mathbf{a}_{t-1}\|_2^2$, $r_{\mathrm{ori},t} = \|\mathbf{g}_{xy,t}\|_2^2$.

Here, $v_{z,t}$ is the vertical base velocity, $\boldsymbol{\omega}_{xy,t}$ is the roll/pitch angular velocity, $\boldsymbol{\tau}_t$ is the joint torque, $\ddot{\mathbf{q}}_t$ is the joint acceleration, $\mathbf{a}_t$ is the current action, and $\mathbf{g}_{xy,t}$ denotes the horizontal components of the projected gravity vector, which are used to penalize non-flat body orientation.

The total reward for PPO training is defined as $r_t = w_{\mathrm{lin}} r_{\mathrm{lin},t} + w_{\mathrm{yaw}} r_{\mathrm{yaw},t} - w_z r_{z,t} - w_\omega r_{\omega xy,t} - w_\tau r_{\tau,t} - w_{\ddot{q}} r_{\ddot{q},t} - w_{\Delta a} r_{\Delta a,t} - w_{\mathrm{ori}} r_{\mathrm{ori},t} + w_{\mathrm{air}} r_{\mathrm{air},t} - w_{\mathrm{contact}} r_{\mathrm{contact},t}$.

The first two terms encourage accurate tracking of the commanded planar velocity and yaw rate. The remaining terms regularize the robot motion by penalizing vertical body motion, roll/pitch rotation, excessive joint torques, large joint accelerations, rapid action changes, undesired contacts, and unstable body orientation. Therefore, the learned low-level motion planner can follow target velocity commands while producing smooth and physically stable quadruped locomotion.

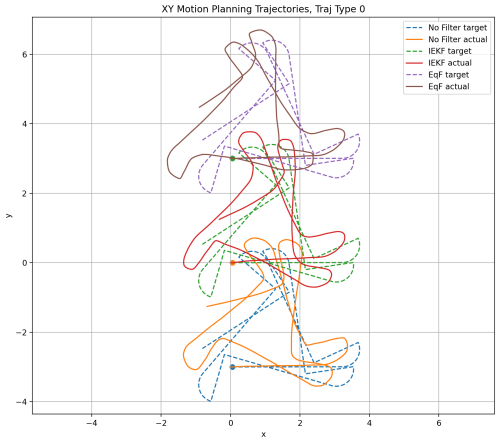
### B.2.3 Low-Level Motion Planner



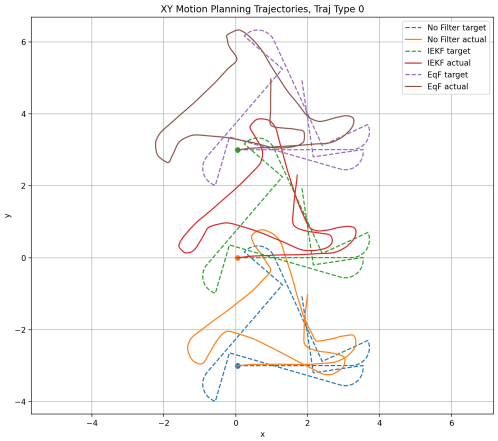
(a) Motion planning Experiments

Figure 8: Visualization of motion planner experiments under noisy observation conditions, comparing the no-filter planner, IEKF-based planner, and EqF-based planner.

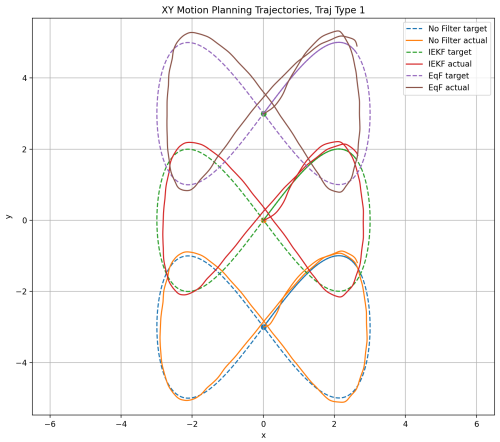
To isolate the role of the motion planner, we design a motion planning experiment under noisy state observations. This experiment does not include pedestrians, as shown in Fig. 8a. Instead, it evaluates whether the low-level motion controller $\pi_l$ can robustly generate joint-level actions when receiving noisy velocity commands. The pretrained Spot locomotion policy $\pi_l$ is used as a fixed low-level controller for all experiments. Given the commanded velocity input, $\pi_l$ outputs 12-dimensional joint



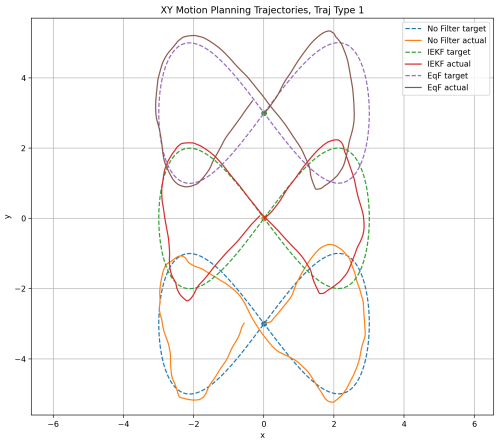

(a) Experiment 1: without noise injection




(b) Experiment 1: with noise injection

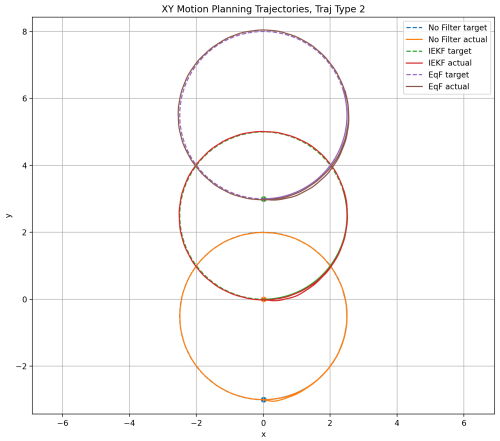



(c) Experiment 2: without noise injection

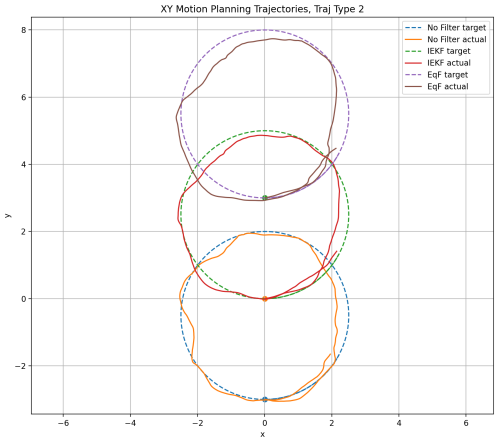



(d) Experiment 2: with noise injection

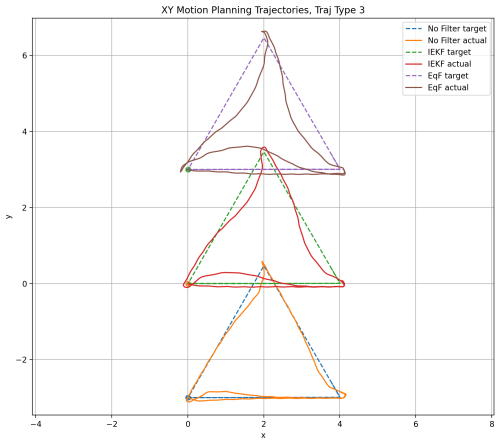
Figure 9: Individual tracking results for Experiments 1 and 2. In each row, the left subfigure shows the tracking result without noise injection, and the right subfigure shows the tracking result with noise injection.



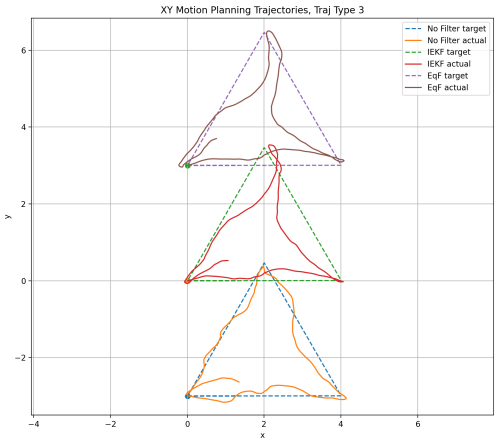

(a) Experiment 3: without noise injection




(b) Experiment 3: with noise injection




(c) Experiment 4: without noise injection




(d) Experiment 4: with noise injection

Figure 10: Individual tracking results for Experiments 3 and 4. In each row, the left subfigure shows the tracking result without noise injection, and the right subfigure shows the tracking result with noise injection.

actions for the quadruped legs. We evaluate its tracking robustness on multiple reference trajectories, including the default combinatorial shape, figure-eight, circular, triangular, and rectangular trajectories.

The robot state is represented as $\chi_t = [p_{x,t}, p_{y,t}, \theta_t, v^B_{x,t}, v^B_{y,t}, \omega^B_{z,t}]^\top$, where $(p_{x,t}, p_{y,t})$ is the global position, $\theta_t$ is the yaw angle, and $(v^B_{x,t}, v^B_{y,t}, \omega^B_{z,t})$ is the body-frame velocity. The noisy measurement is $z_t = \chi_t + n_t$, where $n_t \sim \mathcal{N}(0, R_t)$. The path planner receives the estimated robot state $\hat{\chi}_t$ and a reference pose from the path planner, and outputs a body-frame velocity command $u_t = [v^{cmd}_{x,t}, v^{cmd}_{y,t}, \omega^{cmd}_{z,t}]^\top$.

Given the reference position $p^*_t$ and heading $\theta^*_t$, the planner computes the body-frame tracking error $e^B_{p,t} = R(\hat{\theta}_t)^\top (p^*_t - \hat{p}_t)$ and generates the velocity command:

$$u_t = \begin{bmatrix} v^{*,B}_{x,t} + k_p e^B_{x,t} \\ v^{*,B}_{y,t} + k_p e^B_{y,t} \\ \omega^*_{z,t} + k_\theta \mathrm{wrap}(\theta^*_t - \hat{\theta}_t) \end{bmatrix}. \tag{8}$$

The command is clipped to the robot velocity limits and then passed to the pretrained locomotion policy, which outputs joint-level actions for the quadruped robot.

We compare three planner variants. The no-filter planner directly uses the noisy measurement:

$$\hat{\chi}^{\mathrm{NF}}_t = z_t. \tag{9}$$

The IEKF planner Bonnabel et al. (2009) first predicts the robot state using a command-prior motion model:

$$\bar{v}^B_t = \beta \hat{v}^B_{t|t} + (1-\beta) u_t, \qquad \hat{X}_{t+1|t} = \hat{X}_{t|t} \mathrm{Exp}\left((\bar{v}^B_t \Delta t)^\wedge\right). \tag{10}$$

Then, the IEKF updates the predicted state using the Kalman correction:

$$K_t = P_{t|t-1} H_t^\top \left(H_t P_{t|t-1} H_t^\top + R_t\right)^{-1}, \qquad \hat{\chi}_{t|t} = \hat{\chi}_{t|t-1} + K_t \left(z_t - h(\hat{\chi}_{t|t-1})\right). \tag{11}$$

The EqF planner van Goor et al. (2023) uses the same prediction step, but performs the pose correction directly on the $SE(2)$ group. Given the pose measurement $Z_t \in SE(2)$, the group residual is computed as:

$$r_{X,t} = \mathrm{Log}\left(\hat{X}^{-1}_{t|t-1} Z_t\right). \tag{12}$$

The EqF pose update is then performed through group composition:

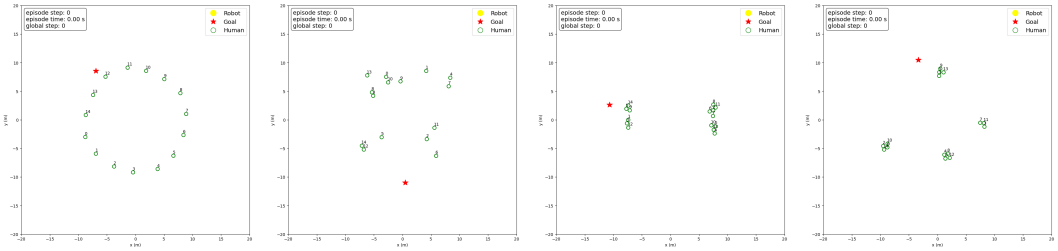
$$\hat{X}_{t|t} = \hat{X}_{t|t-1} \mathrm{Exp}\left(\delta^\wedge_{X,t}\right), \qquad \hat{v}_{t|t} = \hat{v}_{t|t-1} + \delta_{v,t}. \tag{13}$$

All three planners use the same low-level locomotion policy and target trajectories, but differ in the state estimation method used before generating velocity commands. As shown in Fig. 9 and Fig. 10, the motion planning experiments demonstrate that the pretrained policy $\pi_l$ is sufficiently robust to observation noise and can reliably track the target trajectories under both noisy and noise-free conditions.

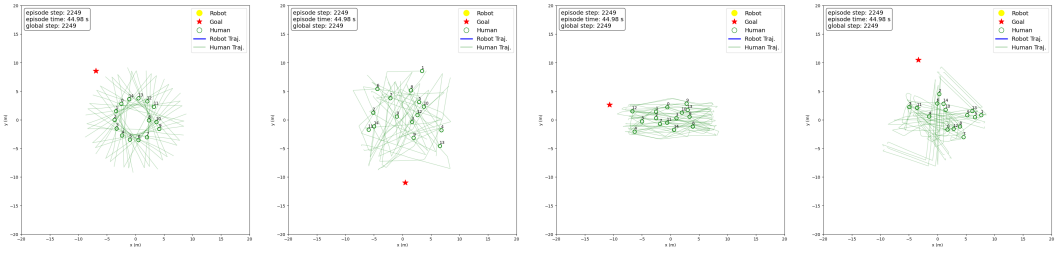
These results indicate that the pretrained locomotion policy is suitable as our low-level motion planner. Although the high-level velocity commands may be affected by noisy state estimation, $\pi_l$ can still accurately execute the commanded motion and produce stable 12-dimensional joint actions for the Spot robot. Therefore, the low-level controller does not become the main bottleneck in the navigation pipeline. Instead, it provides a reliable execution layer that can faithfully track the high-level velocity commands generated by different planners, even in noisy environments. All the results of motion planner trajectory tracking experiments can be accessed by the link: `https://drive.google.com/file/d/17zR5FjnT57wmYvrVxk1tGLHW5Ik1wT0g/view`

### B.3 HUMAN SIMULATOR

#### B.3.1 HUMAN GENERATORS

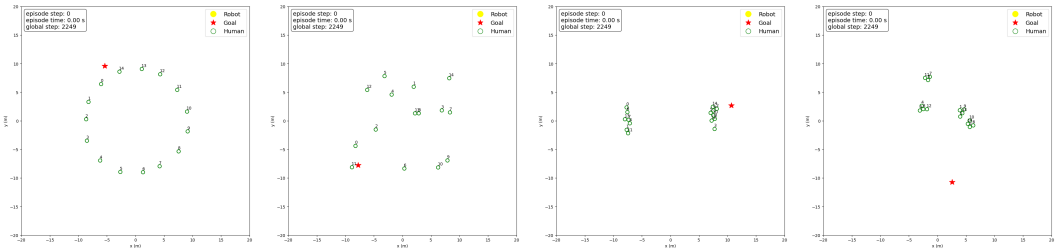


(a) Circle Generation (b) Random Generation (c) Corridor Generation (d) Group Generation

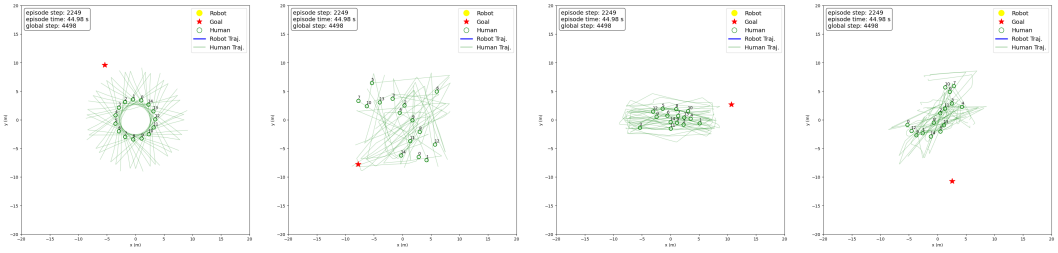


(e) Circle Trajectory (f) Random Trajectory (g) Corridor Trajectory (h) Group Trajectory

Figure 11: Visualization of different human generators in case-1, including circle crossing, random walking, corridor crossing, and group walking scenarios. The first row shows the generated human distributions, while the second row shows the corresponding pedestrian trajectories.



(a) Circle Generation (b) Random Generation (c) Corridor Generation (d) Group Generation



(e) Circle Trajectory (f) Random Trajectory (g) Corridor Trajectory (h) Group Trajectory

Figure 12: Visualization of different human generators in case-2, including circle crossing, random walking, corridor crossing, and group walking scenarios. The first row shows the generated human distributions, while the second row shows the corresponding pedestrian trajectories.

The Human Generator module defines how pedestrians are initialized in each episode, including their starting positions, goal positions, and scenario-level spatial distribution. It is separated from the human motion simulator: the generator decides where humans appear and where they intend

to go, while the simulator, such as Social Force, ORCA, or Human Diffusion, determines how they move over time. This design allows the same navigation policy to be evaluated under different crowd layouts and pedestrian dynamics.

The program supports several procedural human generation modes, including circle-crossing, random start-goal, corridor flow, and group walking, as shown in Fig. 11 and Fig. 12. In the circle-crossing mode, pedestrians are initialized around a circular region and assigned goals on the opposite side, creating dense crossing interactions around the robot. In the random mode, each pedestrian receives an independently sampled start and goal position, which produces diverse open-space social navigation cases. In the corridor mode, pedestrians are distributed along two sides of a corridor-like area and move in opposite directions, forming bidirectional pedestrian flows. In the group mode, pedestrians are divided into small groups, where members of the same group have nearby starts and goals, allowing the environment to model socially coherent group motion.

Several configurable parameters control the behavior of these generators. The random generator includes parameters for the sampling area and the minimum start-goal distance, which determine how widely pedestrians are distributed and how long their intended walking paths are. The corridor generator includes parameters for corridor length, corridor width, and lateral noise, which control the shape and variation of the bidirectional flow. The group generator includes parameters for the number of groups, group spatial spread, goal spread, and minimum group travel distance, which determine the compactness, direction consistency, and movement range of each pedestrian group.

To avoid invalid initialization, the generator also includes a human-human collision checking and correction mechanism. After pedestrian positions are sampled, the environment checks whether any pedestrians overlap or start too close to each other, and then applies an iterative separation process. A minimum robot-human spawning distance is also used to prevent pedestrians from appearing too close to the robot at the beginning of an episode. These mechanisms make the generated scenarios safer, more stable, and more suitable for reinforcement learning.

In addition to procedural generation, the program supports a Human Diffusion mode. In this mode, pedestrian states are not generated by hand-crafted geometric rules. Instead, the environment samples trajectory sequences from a diffusion-based crowd network. This provides more realistic crowd motion patterns and allows the navigation policy to be evaluated under data-driven pedestrian distributions. Invalid or inactive pedestrians are masked or moved away so that they do not affect observation, collision checking, or reward computation.

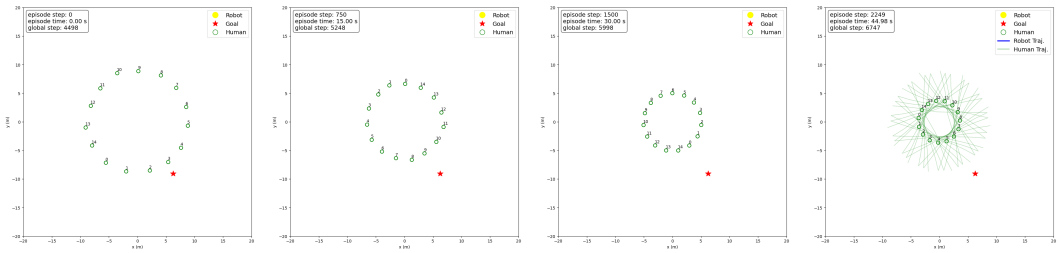
Overall, the human generator module provides diverse crowd initialization conditions for social navigation. The circle mode emphasizes crossing interactions, the random mode provides general open-space encounters, the corridor mode tests navigation in bidirectional pedestrian flows, the group mode evaluates socially aware behavior around cohesive groups, and the Human Diffusion mode introduces realistic data-driven crowd trajectories.

### B.3.2 Human Simulators

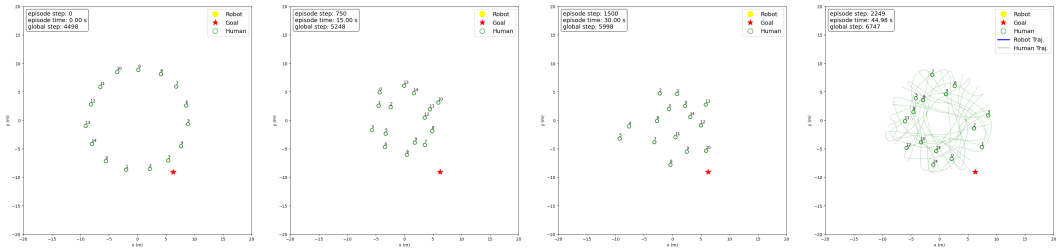
In this program, the pedestrian motion module supports both ORCA and Social Force as rule-based crowd simulators, as shown in Fig. 13 and Fig. 14. The generator defines pedestrian initial states and goals, while ORCA or SF updates pedestrian velocities over time. The simulator can be selected through the human simulator switch.

ORCA is a velocity-based multi-agent collision-avoidance method that allows each agent to independently choose a safe velocity without explicit communication. Its core idea is reciprocal responsibility: when two agents may collide, both share the avoidance effort through small velocity adjustments while still moving toward their goals. For each pedestrian, ORCA predicts possible future collisions with nearby agents, constructs a feasible set of collision-free velocities in velocity space, and selects the velocity closest to the preferred goal-directed velocity. Its main parameters include neighbor detection distance, prediction time horizon, pedestrian radius, safety space, preferred walking speed, and maximum speed, which jointly control sensing range, reaction timing, avoidance conservativeness, and pedestrian motion speed.

The Social Force model describes pedestrian motion as the result of internal "social forces," where each pedestrian is driven toward a desired velocity while being repelled by nearby pedestrians, obstacles, or boundaries. It formulates pedestrian dynamics as nonlinear coupled equations and can

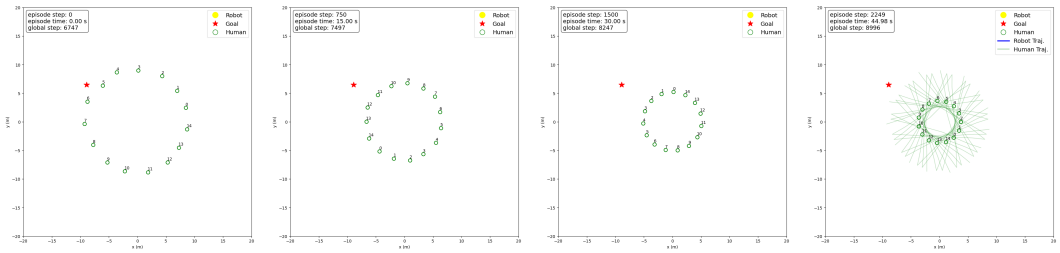


(a) ORCA Step-1 (b) ORCA Step-2 (c) ORCA Step-3 (d) Final Trajectory

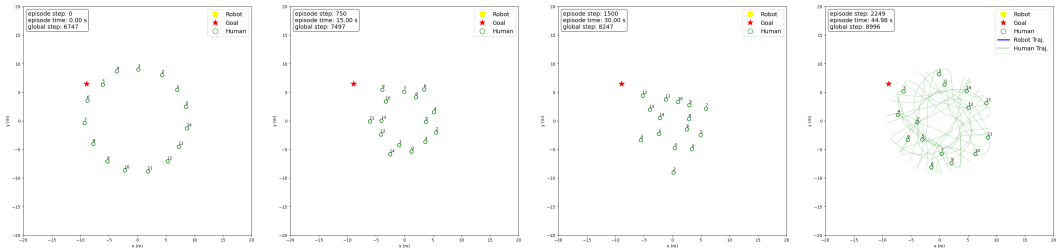


(e) Social Force Step-1 (f) Social Force Step-2 (g) Social Force Step-3 (h) Final Trajectory

Figure 13: Visualization of different human simulators in case-1. The first row shows the crowd evolution generated by the ORCA simulator, while the second row shows the crowd evolution generated by the Social Force simulator.



(a) ORCA Step-1 (b) ORCA Step-2 (c) ORCA Step-3 (d) Final Trajectory



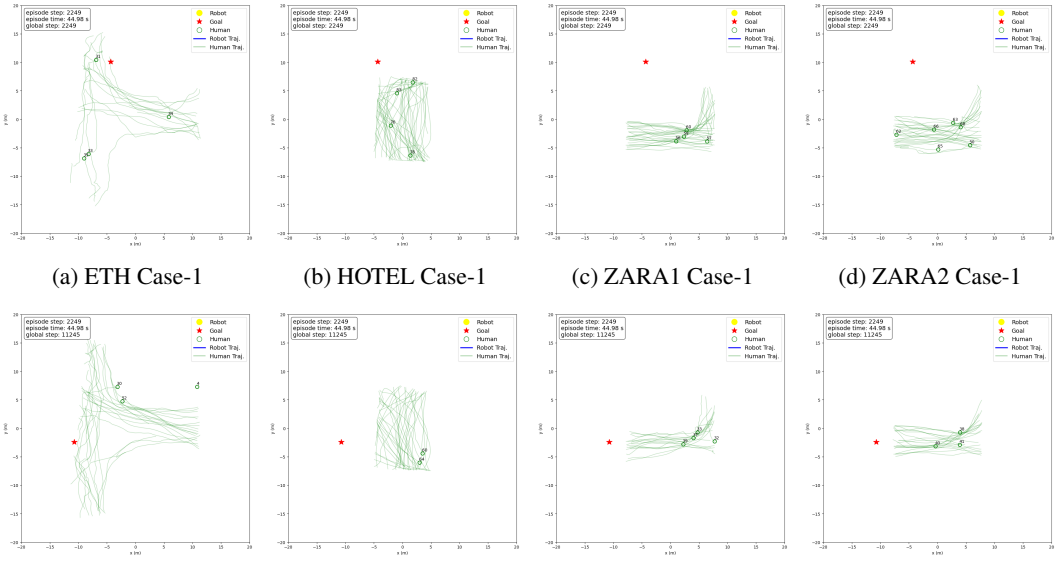
(e) Social Force Step-1 (f) Social Force Step-2 (g) Social Force Step-3 (h) Final Trajectory

Figure 14: Visualization of different human simulators in case-2. The first row shows the crowd evolution generated by the ORCA simulator, while the second row shows the crowd evolution generated by the Social Force simulator.

reproduce collective crowd behaviors. In our program, the SF simulator computes each pedestrian's goal-directed motion, adds repulsive interactions from neighboring humans, and updates the velocity while clipping it within the allowed walking-speed range. Its main parameters include goal-following gain, repulsion strength, repulsion decay range, pedestrian radius, preferred speed, and maximum speed, which jointly control goal tracking, collision avoidance, interaction distance, and crowd movement speed.

Compared with Social Force, ORCA is more dynamic and smooth because motion is generated from direct velocity-space constraints. The Social Force model may suffer from evident motion lag or local stagnation in dense crowds and close-proximity interactions, due to the accumulation of repulsive forces. ORCA, by contrast, generates smoother pedestrian motion through velocity-space collision avoidance, resulting in more continuous and stable crowd dynamics.

### B.3.3 HUMAN DIFFUSION BAE ET AL. (2025)



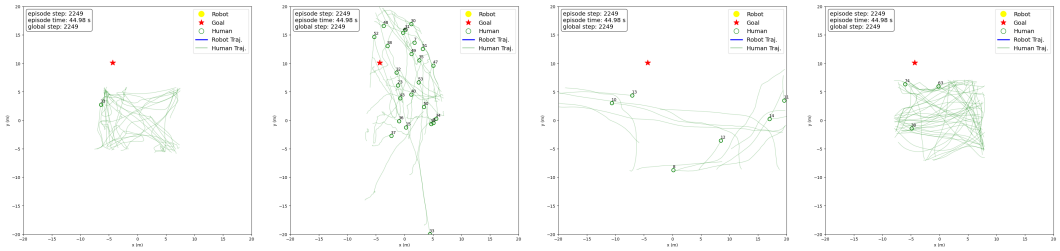
(a) ETH Case-1 (b) HOTEL Case-1 (c) ZARA1 Case-1 (d) ZARA2 Case-1

(e) ETH Case-2 (f) HOTEL Case-2 (g) ZARA1 Case-2 (h) ZARA2 Case-2

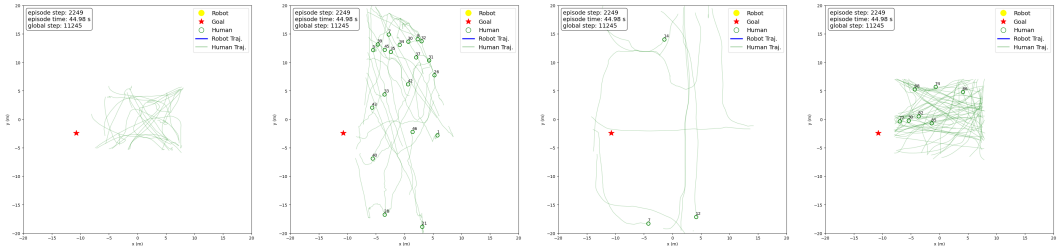
Figure 15: Crowd dynamics visualization on distinct Human Diffusion subdatasets.

In the benchmark, we use the data-driven system of both human generator and human simulator, Human Diffusion, to approximate the real-world human distribution and motion patterns. Human Diffusion generates continuous crowd behaviors by combining a diffusion-based crowd emitter with a state-switching crowd simulator. At the network level, the framework first extracts scene-level spatial conditions from a single image: a pretrained Grounded-SAM model provides semantic segmentation, while a SegFormer-based network predicts the appearance map, population density map, and population probability distribution. The crowd emitter then uses a conditional diffusion model to generate agent parameters, including agent type, walking pace, appearance time, start position, and destination. To capture collective crowd behavior, the diffusion denoising network jointly processes all newly emitted agents with a transformer and uses cross-attention to inject environmental conditions such as spatial layouts and previous crowd occupancy. The crowd simulator further constructs a navigation mesh from traversable regions and uses a Markovian state-switching network to model behavior transitions. Specifically, behavior modes are obtained by clustering normalized trajectory segments, and a transition network predicts the next behavior state conditioned on agent attributes, historical trajectory, neighboring agents, and environmental features. A trajectory decoder then recurrently predicts future locomotion until each agent reaches its destination. This design enables realistic, controllable, and long-horizon crowd generation for social navigation environments.

The crowd emitter uses a diffusion model to generate the initial attributes of pedestrians. For each agent, the parameter vector is defined as $\alpha = \{\kappa, \nu, T_s, c_{T_s}, c_{T_d}\}$, where $\kappa$ denotes the agent type, $\nu$ is the walking pace, $T_s$ is the appearance time, $c_{T_s}$ is the starting position, and $c_{T_d}$ is the destination.

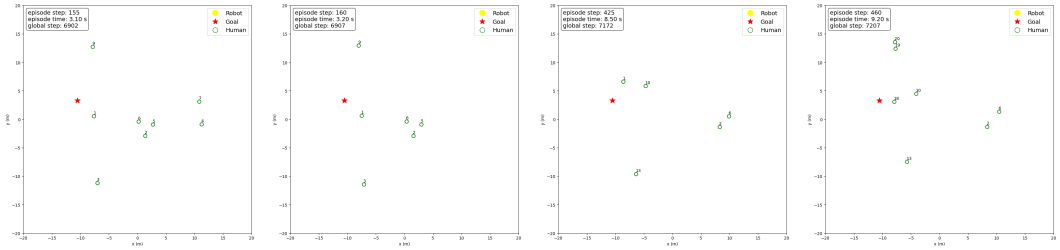


(a) EDIN Case-1 (b) GCS Case-1 (c) SDD Case-1 (d) UNIV Case-1

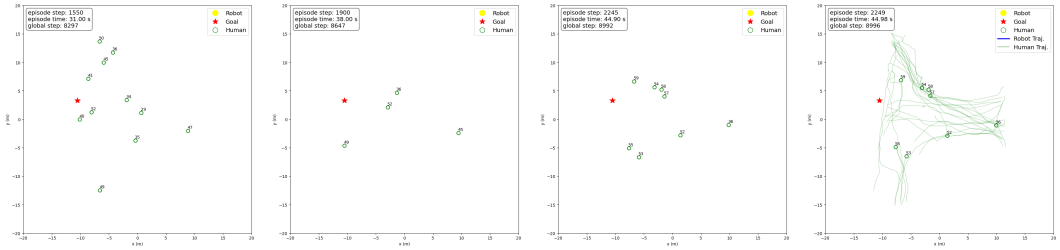


(e) EDIN Case-2 (f) GCS Case-2 (g) SDD Case-2 (h) UNIV Case-2

Figure 16: Crowd dynamics visualization on distinct Human Diffusion subdatasets.



(a) ETH Step-1 (b) ETH Step-2 (c) ETH Step-3 (d) ETH Step-4



(e) ETH Step-5 (f) ETH Step-6 (g) ETH Step-7 (h) Final Trajecotry

Figure 17: Crowd dynamics visualization of the ETH subdataset in a single episode across different time steps.

Instead of directly generating every trajectory point, the diffusion model first generates high-level crowd attributes, including who appears, when the agent appears, where the agent starts, where the agent intends to go, and how fast the agent walks.

The forward diffusion process gradually adds Gaussian noise to the clean agent parameters $\boldsymbol{\alpha}^0$:

$$\begin{aligned} q(\boldsymbol{\alpha}^{1:M} \mid \boldsymbol{\alpha}^0) &= \prod_{m=1}^{M} q(\boldsymbol{\alpha}^m \mid \boldsymbol{\alpha}^{m-1}), \\ q(\boldsymbol{\alpha}^m \mid \boldsymbol{\alpha}^{m-1}) &= \mathcal{N}\left(\boldsymbol{\alpha}^m; \sqrt{1-\beta^m}\boldsymbol{\alpha}^{m-1}, \beta^m \mathbf{I}\right), \end{aligned} \tag{14}$$

where $m$ denotes the diffusion step and $\beta^m$ is the noise scheduling parameter. This process progressively corrupts realistic crowd attributes into Gaussian noise.

The reverse denoising process learns to recover clean agent parameters from noise:

$$\begin{aligned} p_\vartheta(\boldsymbol{\alpha}^{0:M}) &= p(\boldsymbol{\alpha}^M) \prod_{m=1}^{M} p_\vartheta(\boldsymbol{\alpha}^{m-1} \mid \boldsymbol{\alpha}^m), \\ p_\vartheta(\boldsymbol{\alpha}^{m-1} \mid \boldsymbol{\alpha}^m) &= \mathcal{N}\left(\boldsymbol{\alpha}^{m-1}; \boldsymbol{\mu}_\vartheta(\boldsymbol{\alpha}^m, m, \mathbf{C}_e), \beta^m \mathbf{I}\right). \end{aligned} \tag{15}$$

The condition $\mathbf{C}_e$ represents the environmental context $\mathbf{C}_e = \{\mathcal{M}_S, \mathcal{M}_A, \mathcal{M}_P, \mathcal{M}_D\}$, where $\mathcal{M}_S$ is the semantic map, $\mathcal{M}_A$ is the appearance map, $\mathcal{M}_P$ is the population density map, and $\mathcal{M}_D$ is the previous pedestrian position map. Therefore, the diffusion model generates reasonable pedestrian attributes from random noise under scene-level constraints.

After obtaining the starting position and destination of each agent, Human Diffusion uses a navigation mesh to plan feasible walking routes in the environment. The method first removes non-walkable areas, such as buildings, structures, and bushes, according to the semantic segmentation map. Then, it builds a walkable navigation grid and searches for a polyline path from the current position to the destination.

Along this path, the system selects a navigation control point $c_{t,nav}$, which is used to guide the agent's next movement stage. This step enables pedestrians to follow environment-aware routes instead of simply moving in a straight line toward the destination. As a result, the generated trajectories can naturally avoid buildings, obstacles, and complex scene structures.

Another key component of Human Diffusion is the state switching crowd simulator. Instead of assuming that an agent always follows a single motion pattern, Human Diffusion represents pedestrian behavior using multiple discrete behavior states, such as normal walking, avoidance, stopping, turning, and interacting with nearby agents.

The future behavior state is predicted using a Markov-chain style transition model conditioned on historical behavior and environmental context:

$$p_\phi(b_f \mid b_h) = \text{Categorical}\left(b_f; \mu_\phi(b_h, \mathbf{C}_s), b_h\right), \tag{16}$$

where $b_h$ is the historical behavior state, $b_f$ is the future behavior state, and $\mu_\phi$ is the state transition network. The condition $\mathbf{C}_s$ is defined as $\mathbf{C}_s = \{\alpha, \mathbf{c}_{t,nav}, \mathcal{T}_{t-T_h:t}, \mathcal{H}_{t-T_h:t}, \mathcal{M}_S, \mathcal{M}_P\}$.

Here, $\mathcal{T}_{t-T_h:t}$ denotes the historical trajectory of the current agent, and $\mathcal{H}_{t-T_h:t}$ denotes the historical trajectories of neighboring agents. These terms allow the simulator to model human-human interactions and generate more natural behavior transitions, instead of forcing pedestrians to follow a fixed velocity or a fixed goal direction throughout the entire episode.

After sampling the future behavior state $b_f$, Human Diffusion uses a trajectory prediction network to generate the future trajectory over a short time horizon: $\mathcal{T}_{t:t+T_f} = \mu_\varphi(b_f, \mathbf{C}_s)$.

This equation means that, given the future behavior state and contextual information, including environment, history, navigation guidance, and neighboring agents, the network predicts the agent's future positions. By repeatedly applying this process, each pedestrian can gradually move from its starting position to its destination while exhibiting realistic and context-aware crowd behavior.

In the social navigation benchmark, Human Diffusion is used as a data-driven crowd generator and simulator to provide realistic pedestrian distributions and motion patterns, as shown in Fig. 15,

Table 2: Evaluation results of Human Diffusion on crowd subdatasets. ↓ indicates lower is better, while ↑ indicates higher is better.

| Dataset | Scene-Level Realism Metrics | | | | Agent-Level Accuracy Metrics | | | | Detailed Metrics | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Density ↓ | Freq. ↓ | Cov. ↓ | Pop. ↓ | Kinem. ↓ | DTW ↓ | Div. ↑ | Col. ↓ | Vel. ↓ | Acc. ↓ | Dist. ↓ | Time ↓ |
| ETH | 0.01846 | 0.01798 | 0.01798 | 0.18709 | 0.34443 | 1.63282 | 0.19647 | 0.00820 | 0.33425 | 0.33390 | 0.09618 | 0.61338 |
| HOTEL | 0.01092 | 0.00851 | 0.00851 | 0.09638 | 0.32391 | 0.64233 | 0.24556 | 0.01553 | 0.23461 | 0.53921 | 0.09278 | 0.42906 |
| UNIV | 0.34685 | 0.20395 | 0.20395 | 0.73481 | 0.42816 | 1.11499 | 0.34033 | 0.01244 | 0.54897 | 0.55570 | 0.12513 | 0.48284 |
| ZARA1 | 0.01783 | 0.01740 | 0.01740 | 0.26441 | 0.34779 | 0.67640 | 0.30972 | 0.01753 | 0.09221 | 0.90385 | 0.12003 | 0.27506 |
| ZARA2 | 0.00903 | 0.01348 | 0.01348 | 0.09829 | 0.24571 | 0.59110 | 0.35363 | 0.01812 | 0.09073 | 0.51637 | 0.06725 | 0.30848 |
| SDD | 0.18355 | 0.13541 | 0.10347 | 2.17167 | 0.64506 | 5.79248 | 0.31835 | 0.01775 | 0.43120 | 0.98727 | 0.52331 | 0.63847 |
| GCS | 0.24643 | 0.16076 | 0.16076 | 0.78111 | 1.54176 | 4.42777 | 0.30014 | 0.05535 | 0.19726 | 1.45175 | 1.67458 | 2.84344 |
| EDIN | 0.00205 | 0.00183 | 0.00183 | 0.31251 | 0.43687 | 1.30949 | 0.37454 | 0.00000 | 0.27123 | 0.90750 | 0.15081 | 0.41794 |

Table 3: Definitions of the Human Diffusion evaluation metrics. Dens., Freq., Cov., and Pop. are scene-level realism metrics; Kinem., DTW, Div., and Col. are agent-level trajectory metrics. Vel., Acc., Dist., and Time are the four detailed components used in Kinematics.

| Metric | Full Name | Type | Definition | Better |
|---|---|---|---|---|
| Density | Density distribution similarity | Scene-level | Measures the similarity between generated and ground-truth crowd density distributions. The scene is divided into a $10 \times 10$ grid, and the average number of agents per grid cell is compared over time using Earth Mover's Distance (EMD). | ↓ |
| Freq. | Frequency distribution similarity | Scene-level | Measures the similarity of agent-type frequency distributions between generated and ground-truth scenarios. For each grid cell, the number of unique agent types is counted and compared over time using EMD. | ↓ |
| Cov. | Coverage distribution similarity | Scene-level | Measures how similarly the generated and ground-truth crowds occupy the scene. It computes the proportion of occupied grid cells at each time step and compares the temporal coverage distributions using EMD. | ↓ |
| Pop. | Population similarity | Scene-level | Measures the similarity between generated and ground-truth population-size distributions. The number of active agents at each time step is aggregated over the scenario and compared using EMD. | ↓ |
| Kinem. | Kinematics similarity | Agent-level | Measures the similarity of agent kinematic properties between generated and ground-truth trajectories. It is computed as the average of four EMD terms: travel distance, velocity, acceleration, and travel time. | ↓ |
| DTW | Minimum pairwise Dynamic Time Warping | Agent-level | Measures spatial trajectory alignment between generated and ground-truth trajectories. For each trajectory, the closest trajectory in the other set is found using DTW, and the final score averages both generated-to-ground-truth and ground-truth-to-generated directions. The distance is normalized by frame rate. | ↓ |
| Div. | Diversity | Agent-level | Measures trajectory uniqueness/diversity by evaluating how generated trajectories match different ground-truth trajectories, averaged in both generated-to-ground-truth and ground-truth-to-generated directions. | ↑ |
| Col. | Collision rate | Agent-level | Measures the percentage of generated cases where agents collide. A collision is counted when two agents are closer than $0.2$ m. | ↓ |
| Vel. | Travel velocity error | Detailed Kinematics | EMD between generated and ground-truth velocity-profile distributions. | ↓ |
| Acc. | Travel acceleration error | Detailed Kinematics | EMD between generated and ground-truth acceleration-profile distributions. | ↓ |
| Dist. | Travel distance error | Detailed Kinematics | EMD between generated and ground-truth total traveled-distance distributions. | ↓ |
| Time | Travel time error | Detailed Kinematics | EMD between generated and ground-truth travel-duration distributions. | ↓ |

Fig.16 and Fig. 17. To integrate the generated trajectories into the robot-centered navigation task, we further apply robot-centered coordinate normalization, noise perturbation, and scale alignment. Specifically, the human trajectories are centralized with respect to the robot position, and stochastic noise can be injected into pedestrian states to increase motion diversity and avoid deterministic replay. The spatial scale and temporal resolution of the generated trajectories are also aligned with the simulator coordinate system and control frequency, ensuring consistent interaction between the robot and pedestrians. Different from rule-based simulators with a fixed number of always-visible agents, Human Diffusion allows pedestrians to appear and disappear at arbitrary times. Therefore, each pedestrian is represented with a validity mask, so that inactive agents are ignored while active agents participate in observation construction, collision checking, and social interaction evaluation.

### B.4 BASELINES

#### B.4.1 A* HART ET AL. (1968)

We implement A* as a classical receding-horizon heuristic planning baseline for social navigation. At each planning step, the planner receives the current robot state, target position, and surrounding pedestrian states. It constructs a robot-centered local occupancy grid, represents pedestrians as inflated obstacles, and performs A* search from the robot position to the goal or a locally projected goal. The planned path is then tracked through a lookahead waypoint and converted into a robot body-frame velocity command $[v_x, v_y, \omega_z]$.

The local map is represented as a 2D occupancy grid with resolution $r_{\mathrm{grid}}$ and map range $R_{\mathrm{map}}$. In the implementation, the default grid resolution is the parameter $A * -resolution$ within the map range, with the robot located at the grid center. Each pedestrian is modeled as a circular obstacle and inflated by the robot radius, pedestrian radius, and an additional safety margin $r_{\mathrm{inflate}} = r_{\mathrm{robot}} + r_{\mathrm{human}} + r_{\mathrm{safety}}$. With the default parameters, this gives an inflated obstacle radius. Therefore, the planner avoids both pedestrian centers and their surrounding safety buffer regions, encouraging safer navigation in crowded scenarios.

In the occupancy grid, the robot's current position is set as the start node. If the target lies outside the local map boundary, it is first projected into the local map. When either the start node or the goal node is occupied, a breadth-first search is used to find the nearest free cell. After obtaining valid start and goal cells, A* search is performed with the priority function

$$f(n) = g(n) + h(n) r_{\mathrm{grid}}, \tag{17}$$

where $g(n)$ is the accumulated path cost from the start node to node $n$, and $h(n)$ estimates the remaining cost from node $n$ to the goal. We use the Euclidean distance between grid cells as the heuristic:

$$h(n) = \|n - n_{\mathrm{goal}}\|_2 = \sqrt{(i_n - i_g)^2 + (j_n - j_g)^2}, \tag{18}$$

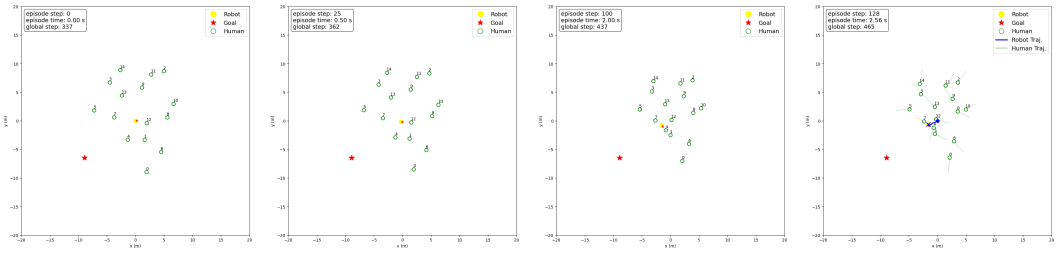
where $n = (i_n, j_n)$ is the current grid cell and $n_{\mathrm{goal}} = (i_g, j_g)$ is the goal cell. The factor $r_{\mathrm{grid}}$ converts the heuristic from grid-cell units to metric distance. When diagonal motion is enabled, the planner uses an 8-connected neighborhood, where horizontal and vertical movements have unit cost and diagonal movements have cost $\sqrt{2}$. Otherwise, a 4-connected grid is used.

After a collision-free path is obtained, the planner does not directly track the entire path. Instead, it selects a local lookahead waypoint as the tracking target. Specifically, the selected waypoint $\mathbf{p}_{\mathrm{wp}}$ is the first waypoint along the planned path whose distance from the robot is larger than the lookahead distance $d_{\mathrm{lookahead}}$. Based on this waypoint, the normalized desired moving direction, the goal-dependent speed scaling factor, and the final desired planar velocity are computed as

$$\mathbf{d}_{\mathrm{wp}} = \frac{\mathbf{p}_{\mathrm{wp}}}{\|\mathbf{p}_{\mathrm{wp}}\|_2}, \; s = \min\left(\frac{d_{\mathrm{goal}}}{r_{\mathrm{slow}}}, 1\right), \; \mathbf{v} = v_{\max}\, s\, \mathbf{d}_{\mathrm{wp}}, \tag{19}$$

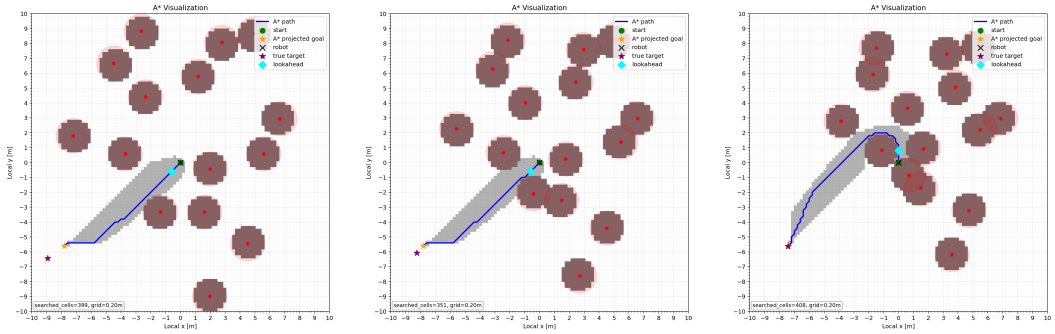
where $\mathbf{p}_{\mathrm{wp}}$ denotes the selected waypoint position in the robot-centered local frame, $d_{\mathrm{goal}}$ is the robot-to-goal distance, $r_{\mathrm{slow}}$ is the slowing radius near the goal, and $v_{\max}$ is the maximum allowed planar speed. The scaling factor $s$ reduces the velocity magnitude when the robot approaches the goal, thereby mitigating overshooting. Finally, the desired world-frame velocity is transformed into the robot body frame and combined with the angular velocity command to form the action.

Overall, this A* planner provides a deterministic classical baseline. It repeatedly rebuilds the occupancy map and replans according to the latest pedestrian positions, enabling reactive obstacle avoidance in dynamic social navigation scenarios.



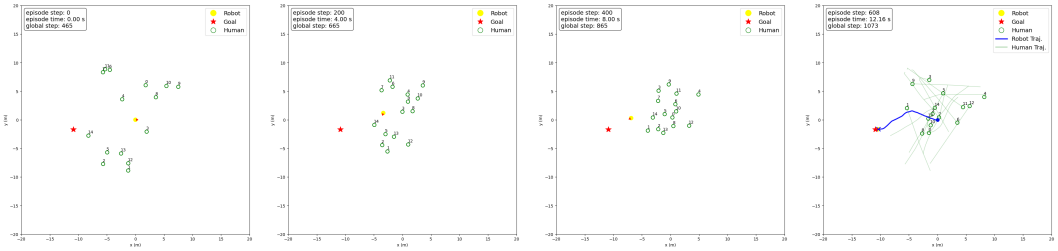
(a) Step 1 (b) Step 2 (c) Step 3 (d) Trajectory [Collision]

Figure 18: A* trajectory planning procedure in Case 1.



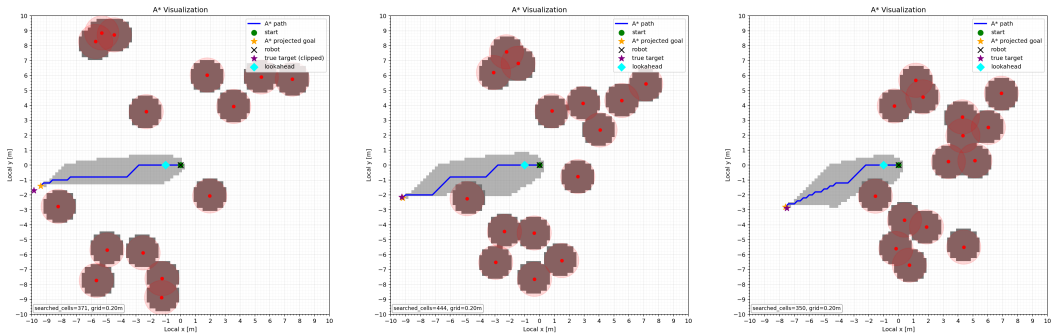
(a) A* step 1 (b) A* step 2 (c) A* step 3

Figure 19: A* path planning visualization in case-1.



(a) Step 1 (b) Step 2 (c) Step 3 (d) Trajectory [Success]

Figure 20: A* trajectory planning procedure in Case 2.



(a) A* step 1 (b) A* step 2 (c) A* step 3

Figure 21: A* path planning visualization in case-2.

### B.4.2 RRT* KARAMAN & FRAZZOLI (2011)

The RRT* baseline is implemented as a receding-horizon sampling-based planner. At each planning step, the robot constructs a bounded local continuous planning space centered at its current position, projects the global target into this local region, and represents nearby pedestrians as inflated circular obstacles. A rapidly exploring random tree is incrementally grown from the robot position by sampling collision-free states, steering toward sampled points with a fixed step size, and rejecting nodes or edges that intersect inflated human obstacles. Compared with standard RRT, RRT* further optimizes the tree by selecting the minimum-cost parent from nearby nodes and rewiring neighboring branches when a lower-cost connection is found. Once a collision-free connection to the projected goal is available, the path is recovered by backtracking the parent links. The robot then follows the path using a lookahead waypoint strategy, converting the selected waypoint into a smooth velocity command. This design allows RRT* to serve as a traditional sampling-based navigation baseline under dynamic human obstacles.

RRT* incrementally builds a search tree from the current robot position. The root node is initialized as the robot position in the local planning frame, denoted as $\mathcal{T} = \{p_{\text{start}}\}$. At each iteration, the planner samples a candidate point from the local continuous planning space. With probability $p_{\text{goal}}$, the projected goal $p_g$ is sampled to bias the tree expansion toward the target; otherwise, a random point $p_{\text{rand}}$ is sampled from the local space. Therefore, the sampled point can be written as $p_{\text{sample}} = p_g$ with probability $p_{\text{goal}}$, and $p_{\text{sample}} = p_{\text{rand}}$ otherwise. Here, $p_{\text{goal}}$ corresponds to the goal sampling rate in the implementation.

Given the sampled point, RRT* first finds the nearest node in the existing tree, computed as $p_{\text{near}} = \arg\min_{p_i \in \mathcal{T}} \|p_i - p_{\text{sample}}\|$. A new node is then generated by steering from $p_{\text{near}}$ toward $p_{\text{sample}}$ with a bounded step size. Specifically, the new node is computed as

$$p_{\text{new}} = p_{\text{near}} + \min(\texttt{step_size}, \|p_{\text{sample}} - p_{\text{near}}\|) \frac{p_{\text{sample}} - p_{\text{near}}}{\|p_{\text{sample}} - p_{\text{near}}\|}. \tag{20}$$

The new node is accepted only if both the node itself and the edge connecting it to its parent are collision-free with respect to all inflated human obstacles.

Unlike vanilla RRT, RRT* improves path quality by optimizing the tree structure during expansion. After a valid node $p_{\text{new}}$ is generated, the planner searches for nearby tree nodes within a predefined search radius, denoted as

$$\mathcal{N}(p_{\text{new}}) = \{p_i \in \mathcal{T} \mid \|p_i - p_{\text{new}}\| \le r_{\text{search}}\}. \tag{21}$$

Among these nearby nodes, RRT* selects the feasible parent that gives the minimum accumulated path cost, written as:

$$p_{\text{parent}} = \arg \min_{p_i \in \mathcal{N}(p_{\text{new}})} (J(p_i) + \|p_i - p_{\text{new}}\|). \tag{22}$$

Where $J(p_i)$ denotes the accumulated path cost from the root node to node $p_i$.

After adding $p_{\text{new}}$ to the tree, RRT* further checks whether the nearby nodes can obtain a lower-cost connection through $p_{\text{new}}$. If the new connection reduces the accumulated cost and the corresponding edge is collision-free, the parent of that nearby node is updated. This rewiring process is the key mechanism that allows RRT* to gradually improve the path quality compared with standard RRT.

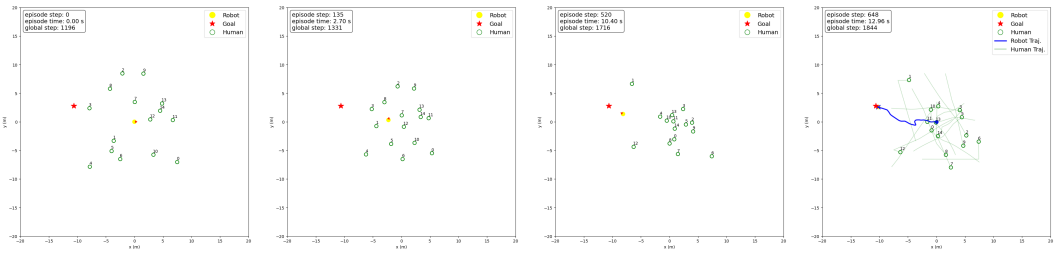
During tree expansion, if a newly added node can be connected to the projected goal without collision, the planner computes the corresponding goal connection cost as:

$$J_{\text{goal}} = J(p_{\text{new}}) + \|p_{\text{new}} - p_g\|. \tag{23}$$

The node with the lowest valid goal connection cost is stored as the current best goal parent. After the search terminates, if such a goal connection exists, the final path is recovered by backtracking the parent links from the goal-connected node to the start node.

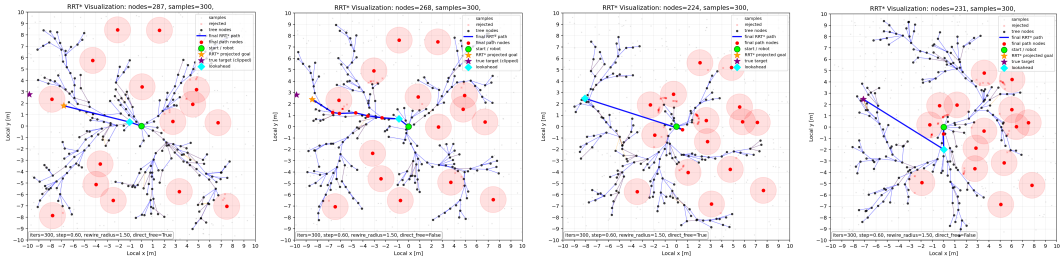
If no collision-free connection to the projected goal is found, the planner does not stop directly. Instead, it selects the tree node that is closest to the projected goal and reconstructs the corresponding branch as a fallback path. This allows the robot to still move toward a locally feasible direction even when a complete path to the goal is not found within the current planning horizon.

After the RRT* path is obtained, the robot does not directly track the full path. Instead, it uses a lookahead waypoint tracking strategy. The planner selects the first waypoint on the path whose distance from the robot is larger than the lookahead distance, denoted as $p_{\text{wp}}$, where $\|p_{\text{wp}}\| \geq d_{\text{lookahead}}$. The selected waypoint defines the desired moving direction as $\hat{v} = p_{\text{wp}}/\|p_{\text{wp}}\|$. To avoid overshooting near the goal, a speed reduction factor is applied according to the current goal distance, computed as $s = \min(d_{\text{goal}}/r_{\text{slow}}, 1)$. The final desired world-frame velocity is then computed as $v_{\text{world}} = v_{\max} \cdot s \cdot \hat{v}$. This velocity is finally converted into the robot body-frame action command $[v_x, v_y, \omega_z]$ for execution by the low-level controller.



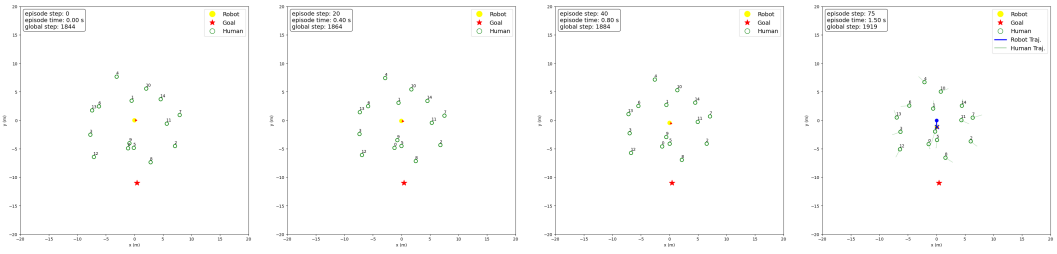
(a) Step 1 (b) Step 2 (c) Step 3 (d) Trajectory [Success]

Figure 22: RRT trajectory planning procedure in Case 1.



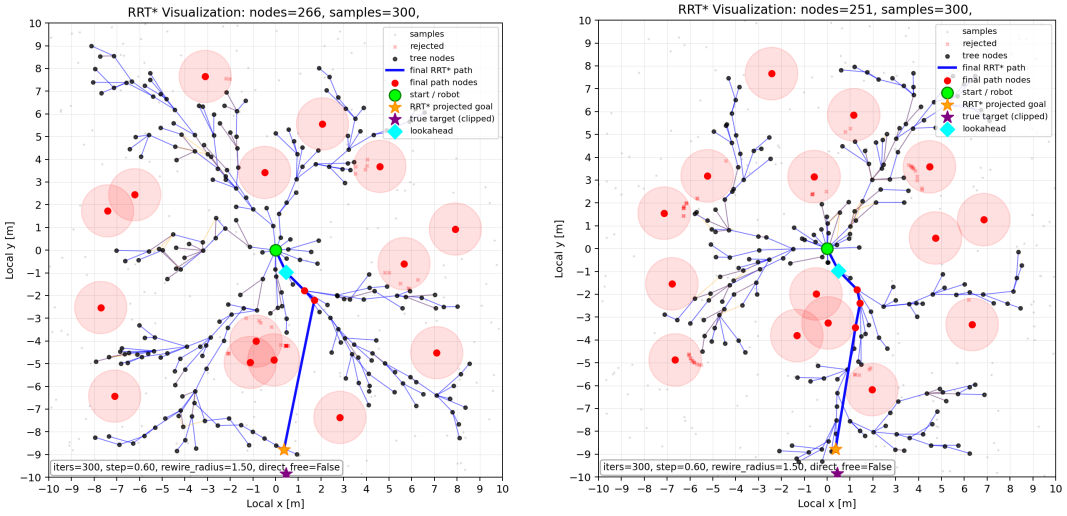
(a) RRT step 1 (b) RRT step 2 (c) RRT step 3 (d) RRT step 4

Figure 23: RRT path planning visualization in Case 1.



(a) Step 1 (b) Step 2 (c) Step 3 (d) Trajectory [Collision]

Figure 24: RRT trajectory planning procedure in Case 2.




(a) RRT step 1

(b) RRT step 2

Figure 25: RRT path planning visualization in Case 2.

### B.4.3 SOCIAL FORCE HELBING & MOLNAR (1995)

The Social Force (SF) is a classical microscopic pedestrian motion model that describes human movement as the result of several virtual forces. Instead of directly specifying pedestrian trajectories, SFM assumes that each pedestrian is driven by an internal motivation to move toward the desired goal while being repelled by other pedestrians and obstacles. Therefore, the motion of each pedestrian is determined by the combination of a goal-directed driving force and interaction forces from the surrounding environment.

For each pedestrian $i$, we denote its position, velocity, desired speed, and desired walking direction as $\mathbf{p}_i$, $\mathbf{v}_i$, $v_i^0$, and $\mathbf{e}_i$, respectively. The overall motion is governed by a force-based acceleration model:

$$\frac{d\mathbf{v}_i}{dt} = \mathbf{f}_i^{\text{goal}} + \sum_{j \neq i} \mathbf{f}_{ij}^{\text{ped}} + \sum_o \mathbf{f}_{io}^{\text{obs}}, \tag{24}$$

where $\mathbf{f}_i^{\text{goal}}$ is the driving force toward the pedestrian's goal, $\mathbf{f}_{ij}^{\text{ped}}$ is the repulsive interaction force from another pedestrian $j$, and $\mathbf{f}_{io}^{\text{obs}}$ is the repulsive force from an obstacle or boundary.

The goal-directed driving force encourages each pedestrian to move toward its desired velocity:

$$\mathbf{f}_i^{\text{goal}} = \frac{v_i^0 \mathbf{e}_i - \mathbf{v}_i}{\tau_i}, \tag{25}$$

where $\tau_i$ is the relaxation time that controls how quickly the pedestrian adapts its current velocity to the desired velocity.

The pedestrian-pedestrian interaction force is commonly modeled as a distance-dependent repulsive force:

$$\mathbf{f}_{ij}^{\text{ped}} = A_i \exp\left(\frac{r_{ij} - d_{ij}}{B_i}\right) \mathbf{n}_{ij}, \tag{26}$$

where $A_i$ controls the interaction strength, $B_i$ controls the interaction range, $r_{ij} = r_i + r_j$ is the sum of pedestrian radii, $d_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|$ is the distance between pedestrians, and $\mathbf{n}_{ij}$ is the normalized direction from pedestrian $j$ to pedestrian $i$.

Similarly, the obstacle repulsive force can be written as

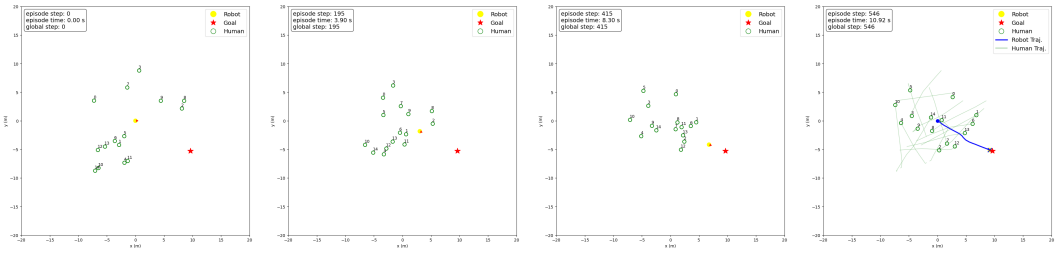
$$\mathbf{f}_{io}^{\text{obs}} = A_o \exp\left(\frac{r_i - d_{io}}{B_o}\right) \mathbf{n}_{io}, \tag{27}$$

where $d_{io}$ is the shortest distance from pedestrian $i$ to obstacle $o$, and $\mathbf{n}_{io}$ denotes the repulsive direction from the obstacle to the pedestrian.

After computing the total force, the pedestrian velocity and position are updated by numerical integration:

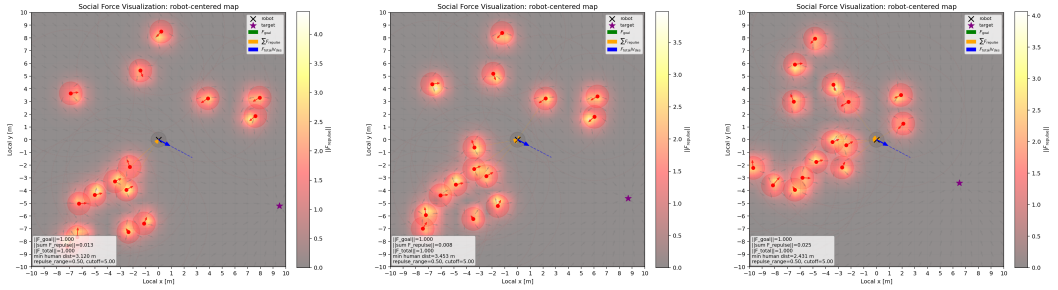
$$\mathbf{v}_i^{t+\Delta t} = \mathbf{v}_i^t + \Delta t \left(\mathbf{f}_i^{\text{goal}} + \sum_{j \neq i} \mathbf{f}_{ij}^{\text{ped}} + \sum_o \mathbf{f}_{io}^{\text{obs}}\right), \ \mathbf{p}_i^{t+\Delta t} = \mathbf{p}_i^t + \Delta t \mathbf{v}_i^{t+\Delta t}. \tag{28}$$

In social navigation, the robot can be treated as a dynamic obstacle or an additional interacting agent. Pedestrians are driven toward their own goals while being repelled by nearby pedestrians, obstacles, and the robot. Compared with ORCA, which constructs explicit velocity-space constraints, the Social Force Model generates crowd motion through continuous interaction forces. This makes it suitable for simulating smooth and reactive pedestrian behaviors in robot navigation environments.



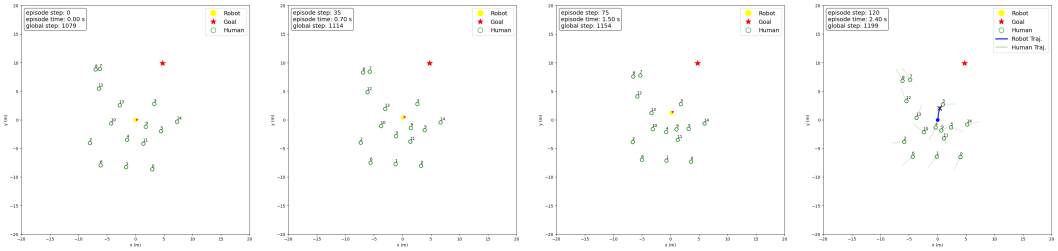
(a) Step 1 (b) Step 2 (c) Step 3 (d) Trajectory [Success]

Figure 26: Social Force path planning procedure in Case 1.



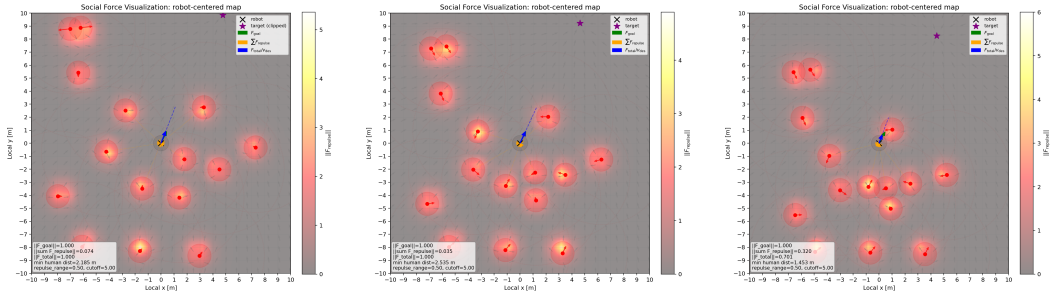
(a) Social Force step 1 (b) Social Force step 2 (c) Social Force step 3

Figure 27: Social Force path planning visualization in Case 1.



(a) Step 1 (b) Step 2 (c) Step 3 (d) Trajectory [Collision]

Figure 28: Social Force path planning procedure in Case 2.



(a) Social Force step 1 (b) Social Force step 2 (c) Social Force step 3

Figure 29: Social Force path planning visualization in Case 2.

### B.4.4 ORCA ALONSO-MORA ET AL. (2013)

Optimal Reciprocal Collision Avoidance (ORCA) is a velocity-based multi-agent collision avoidance method. Its key idea is reciprocal responsibility: when two agents may collide within a future time horizon, both agents share the avoidance responsibility instead of requiring only one agent to react. Therefore, each agent only needs to apply a partial velocity correction, enabling distributed, smooth, and real-time local collision avoidance.

For an agent $A$, we denote its position, velocity, radius, and preferred velocity as $\mathbf{p}_A$, $\mathbf{v}_A$, $r_A$, and $\mathbf{v}_{\text{pref}}$, respectively. For a neighboring agent $B_i$, the relative position, relative velocity, and combined radius are defined as $\mathbf{p}_{AB_i} = \mathbf{p}_{B_i} - \mathbf{p}_A, \quad \mathbf{v}_{AB_i} = \mathbf{v}_A - \mathbf{v}_{B_i}, \quad R_{AB_i} = r_A + r_{B_i}$.

ORCA first constructs a velocity obstacle within a prediction time horizon $\tau$. The velocity obstacle represents the set of relative velocities that would lead agent $A$ to collide with agent $B_i$ within the time interval $[0, \tau]$:

$$VO^{\tau}_{A|B_i} = \{\mathbf{v} \mid \exists t \in [0,\tau],\ \|\mathbf{p}_{AB_i} - t\mathbf{v}\| < R_{AB_i}\}. \tag{29}$$

If the current relative velocity $\mathbf{v}_{AB_i}$ lies inside $VO^{\tau}_{A|B_i}$, then a potential collision may occur within the time horizon $\tau$. ORCA computes the minimum velocity correction $\mathbf{u}_{AB_i}$ that moves the current relative velocity outside the velocity obstacle:

$$\mathbf{u}_{AB_i} = \arg\min_{\mathbf{u}} \|\mathbf{u}\|, \quad \text{s.t.} \quad \mathbf{v}_{AB_i} + \mathbf{u} \notin VO^{\tau}_{A|B_i}. \tag{30}$$

Under the reciprocal collision avoidance assumption, agents $A$ and $B_i$ each take half of the responsibility for avoiding the collision. Thus, agent $A$ only applies half of the required velocity correction $\mathbf{v}_A^{\text{boundary}} = \mathbf{v}_A + \frac{1}{2}\mathbf{u}_{AB_i}$.

Based on this boundary velocity, the ORCA half-plane constraint induced by neighbor $B_i$ is defined as

$$ORCA^{\tau}_{A|B_i} = \left\{\mathbf{v} \;\middle|\; \left(\mathbf{v} - \left(\mathbf{v}_A + \frac{1}{2}\mathbf{u}_{AB_i}\right)\right) \cdot \mathbf{n}_{AB_i} \geq 0\right\}, \tag{31}$$

where $\mathbf{n}_{AB_i}$ is the outward normal direction of the velocity obstacle boundary $\mathbf{n}_{AB_i} = \frac{\mathbf{u}_{AB_i}}{\|\mathbf{u}_{AB_i}\|}$.

When multiple neighboring agents are present, each neighbor introduces one ORCA constraint. The feasible safe velocity set for agent $A$ is the intersection of all ORCA half-plane constraints:

$$\mathcal{V}_A^{\text{safe}} = \bigcap_{B_i \in \mathcal{N}_A} ORCA^{\tau}_{A|B_i}, \tag{32}$$

where $\mathcal{N}_A$ denotes the set of neighboring agents around $A$.

Finally, ORCA selects the feasible velocity that is closest to the preferred goal-directed velocity:

$$\mathbf{v}_{\text{ORCA}} = \arg\min_{\mathbf{v}} \|\mathbf{v} - \mathbf{v}_{\text{pref}}\|, \quad \text{s.t.} \quad \mathbf{v} \in \bigcap_{B_i \in \mathcal{N}_A} ORCA^{\tau}_{A|B_i}. \tag{33}$$

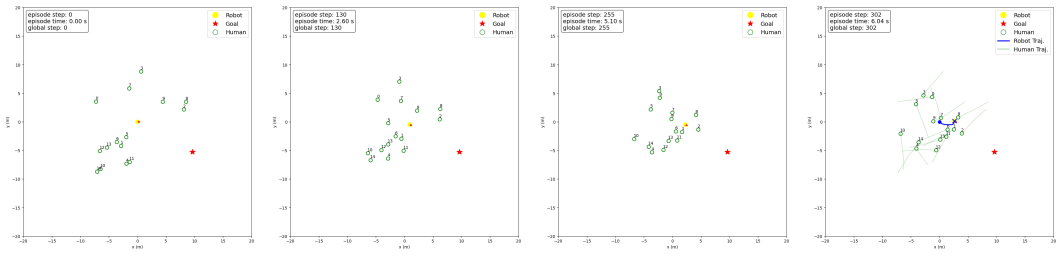
Equivalently, the optimization can be written as

$$\mathbf{v}_{\text{ORCA}} = \arg \min_{\mathbf{v} \in \bigcap_{B_i \in \mathcal{N}_A} ORCA^{\tau}_{A|B_i}} \|\mathbf{v} - \mathbf{v}_{\text{pref}}\|. \tag{34}$$

In social navigation, the robot is treated as agent $A$, while surrounding pedestrians are treated as neighboring agents $B_i$. The robot preferred velocity is usually generated from the direction toward the goal:

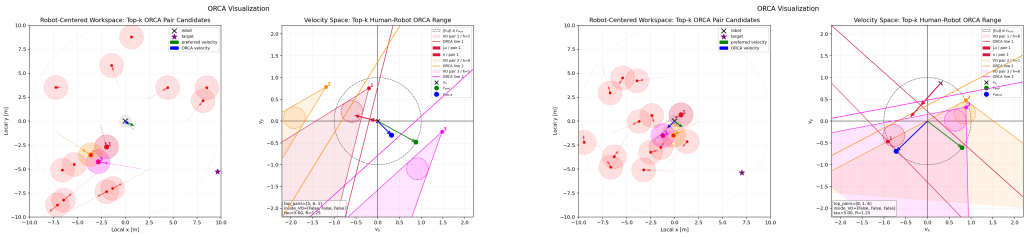
$$\mathbf{v}_{\text{pref}} = v_{\max} \frac{\mathbf{p}_g - \mathbf{p}_A}{\|\mathbf{p}_g - \mathbf{p}_A\|}, \tag{35}$$

where $\mathbf{p}_g$ is the goal position and $v_{\max}$ is the maximum speed. Therefore, ORCA computes a locally safe velocity that remains as close as possible to the goal-directed velocity while satisfying all reciprocal collision-avoidance constraints induced by nearby pedestrians.



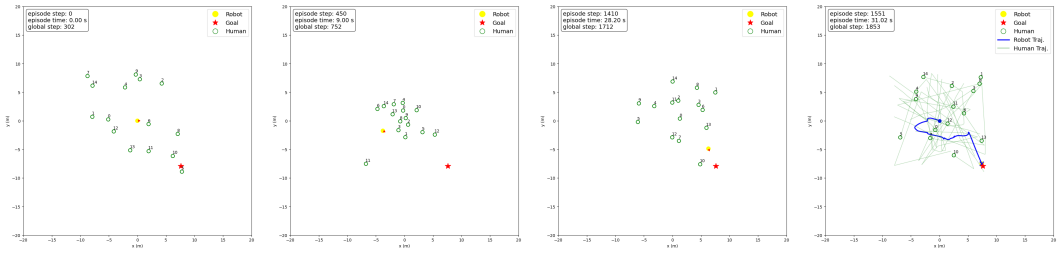
(a) Step 1 (b) Step 2 (c) Step 3 (d) Trajectory [Collision]

Figure 30: ORCA trajectory planning procedure in Case 1.



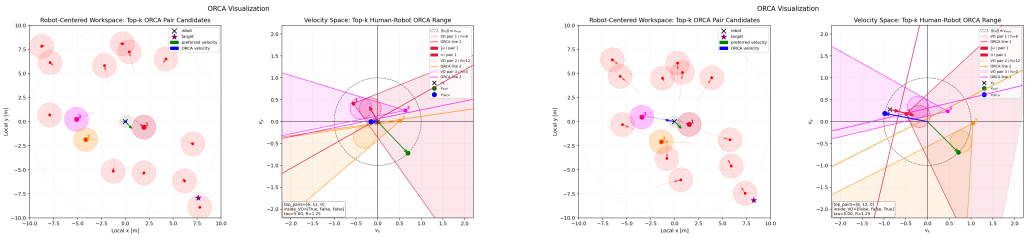
(a) ORCA step 1 (b) ORCA step 2

Figure 31: ORCA path planning visualization in Case 1.



(a) Step 1 (b) Step 2 (c) Step 3 (d) Trajectory [Success]

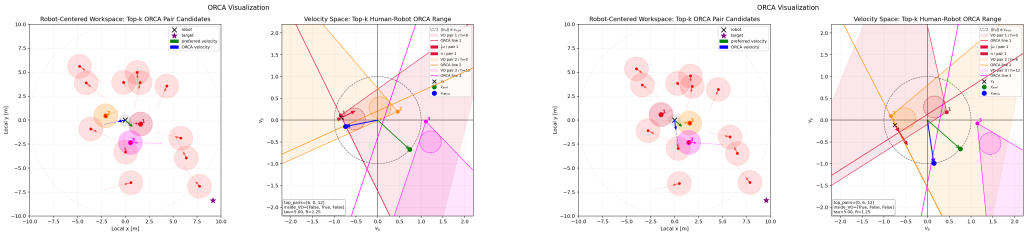
Figure 32: ORCA trajectory planning procedure in Case 2.



(a) ORCA step 1



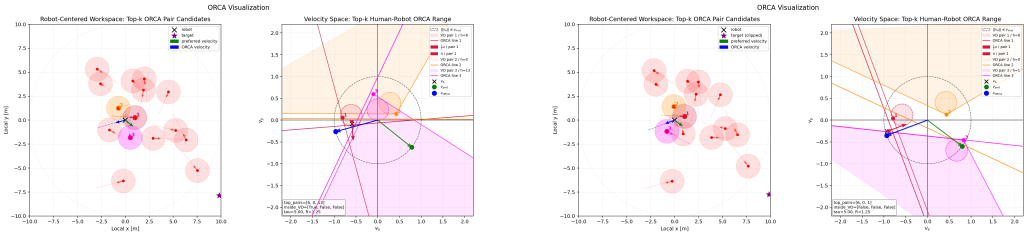
(b) ORCA step 2



(c) ORCA step 3



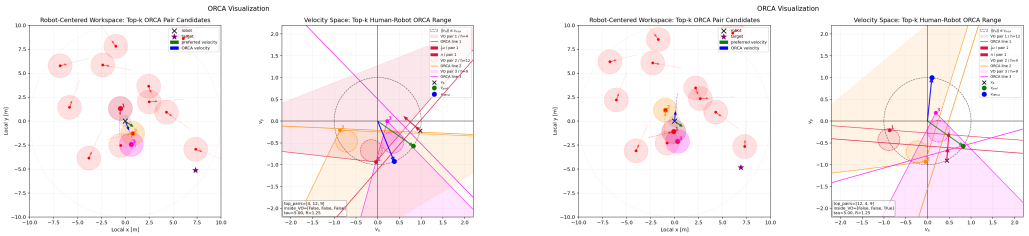
(d) ORCA step 4



(e) ORCA step 5



(f) ORCA step 6



(g) ORCA step 7



(h) ORCA step 8

Figure 33: ORCA path planning visualization in Case 2.

#### B.4.5 CADRL CHEN ET AL. (2017)

Collision Avoidance with Deep Reinforcement Learning (CADRL) is a decentralized learning-based collision avoidance method that models robot-agent interactions through pairwise value estimation. In our benchmark, we adapt CADRL as a PPO-compatible human-robot interaction encoder rather than directly using its original action-selection mechanism. For each robot-human pair, we construct a 13-dimensional goal-aligned state containing the robot-goal distance, preferred speed, heading, robot radius and velocity, the relative human position and velocity, the human radius, the robot-human distance, and the radius sum. This state is rotated into the robot-goal coordinate frame to remove dependence on the global scene orientation.

Each pairwise state is encoded by an MLP value network, producing both a latent pair feature and a scalar pair value. For multi-human aggregation, we follow the conservative CADRL principle by selecting the human with the minimum pair value, which corresponds to the most critical interaction. The selected pair feature is used as the PPO actor representation, while the critic either directly uses the minimum pair value or an optional refined critic head. The final policy is trained end-to-end with PPO in the IsaacSim social navigation benchmark and outputs continuous velocity commands. This CADRL-PPO baseline therefore preserves the pairwise value-based collision avoidance structure of CADRL while enabling fair comparison with other PPO-trained navigation policies under the same observation, reward, and evaluation protocol.

#### B.4.6 LSTM-RL EVERETT ET AL. (2018), SARL CHEN ET AL. (2019)

**LSTM-RL.** LSTM-RL is a recurrent reinforcement learning baseline for crowd navigation. Different from CADRL, which mainly selects the most critical robot-human pair, LSTM-RL uses a recurrent encoder to aggregate information from all surrounding pedestrians. In this way, the robot can reason about the crowd as a sequence of human-robot interaction features.

In our benchmark, LSTM-RL is implemented as a PPO-compatible actor-critic backbone. We first construct a CADRL-style 13-dimensional pairwise state for each robot-human pair, including the robot-goal distance, preferred speed, heading, robot velocity, relative human position and velocity, human radius, robot-human distance, and radius sum. The state is rotated into the robot-goal coordinate frame to remove dependence on the global scene orientation.

Each pairwise state is encoded by an MLP, and the resulting human interaction features are passed into an LSTM. The final hidden state of the LSTM is used as the crowd-level representation. This crowd feature is concatenated with the robot self-state and then processed by actor and critic networks. The actor outputs the continuous robot command, while the critic estimates the PPO value function. Therefore, LSTM-RL serves as a recurrent multi-human interaction baseline under the same observation, reward, action space, and evaluation protocol as the other methods.

**SARL.** Socially Attentive Reinforcement Learning (SARL) is an attention-based reinforcement learning baseline for social navigation. Its main idea is that not all pedestrians contribute equally to the robot's decision. Instead of simply pooling all human features, SARL learns attention weights over surrounding humans and focuses more on the pedestrians that are more relevant to the current navigation situation.

In our benchmark, SARL is also adapted as a PPO-compatible actor-critic backbone. Similar to LSTM-RL and CADRL, SARL first builds a 13-dimensional rotated pairwise state for each robot-human pair. Each pairwise state is encoded by an MLP to obtain a human-robot interaction feature. Then, SARL computes an attention score for each human. When the global-state option is enabled, the attention module also considers the average crowd feature, allowing the model to reason about both individual pedestrians and the overall crowd context.

The final crowd representation is obtained by attention-weighted aggregation of all human interaction features. This representation is concatenated with the robot self-state and passed into the PPO actor and critic. The actor generates the continuous velocity command, and the critic predicts the state value. In this way, SARL preserves its original social-attention mechanism while being trained end-to-end with PPO in our benchmark.

### B.4.7 DSRNN Liu et al. (2021)

Deep Structured Recurrent Neural Network (DSRNN) is a recurrent social navigation baseline that represents the robot and surrounding humans as a spatio-temporal interaction graph. In this graph, the robot is modeled as a node, while temporal edges describe the robot's motion history and spatial edges describe the interactions between the robot and nearby pedestrians. Compared with CADRL, LSTM-RL, and SARL, DSRNN explicitly maintains recurrent hidden states for both node dynamics and edge dynamics, making it suitable for modeling time-dependent crowd interactions.

In our benchmark, we adapt DSRNN as a PPO-compatible recurrent actor-critic baseline. The input observation contains the robot node state, temporal edge features, and spatial edge features. The temporal edge RNN encodes the robot's motion dynamics across time, while the spatial edge RNN encodes the pairwise robot-human interaction features. An edge-attention module is then used to aggregate the spatial edge hidden states, allowing the robot to focus on the most relevant pedestrians in the current scene.

The attended spatial-edge feature is combined with the temporal-edge feature and the robot node embedding, and then passed into the human-node RNN. The output of the node RNN is used as the final social interaction representation for PPO. The actor network maps this representation to the robot action feature, while the critic network estimates the state value. The final policy outputs the same continuous velocity command as the other baselines.

Therefore, DSRNN is used in our IsaacSim social navigation benchmark as a recurrent graph-based baseline. It differs from LSTM-RL, which sequentially aggregates human features, and SARL, which performs attention over independent pairwise features. Instead, DSRNN models social navigation through structured node and edge recurrent modules, preserving both temporal motion information and spatial human-robot interaction information. This allows it to be fairly compared with CADRL, LSTM-RL, SARL, NaviSTAR, NaviDIFF, and NaviVLA under the same PPO training pipeline, reward function, action space, and evaluation protocol.

### B.4.8 NaviSTAR Wang et al. (2023)

NaviSTAR is a transformer-based social navigation baseline that models both spatial crowd interactions and temporal motion history. Different from recurrent methods such as LSTM-RL and DSRNN, NaviSTAR uses attention modules to capture long-range dependencies across time and across surrounding agents. It combines graph-based spatial reasoning with transformer-based temporal modeling, making it suitable for dynamic crowd navigation.

In our benchmark, NaviSTAR is adapted as a PPO-compatible actor-critic backbone. The input observation contains the robot node state, temporal edge features, spatial edge features, visibility masks, and robot position. The robot state is first embedded into a latent feature space. Meanwhile, the robot and surrounding humans are treated as nodes in a spatial interaction graph. The graph adjacency is constructed from pairwise distances between agents, and a GCN module is used to encode local spatial interaction structure.

To further model social interactions, NaviSTAR applies a spatial transformer over the human and robot spatial-edge features. A cross-attention module then uses the graph-based robot feature as the query and the spatial transformer features as keys and values. This allows the robot representation to selectively attend to important surrounding agents while considering the learned spatial graph structure.

The resulting social interaction feature is fused with the robot node embedding. A fusion transformer is then applied along the temporal dimension to model the history of robot-crowd interactions. The final output feature is used by separate PPO actor and critic heads. The actor generates the continuous robot command, while the critic estimates the state value.

Therefore, NaviSTAR serves as a spatial-temporal transformer baseline in our IsaacSim social navigation benchmark. Compared with SARL, which performs attention over humans at a single timestep, NaviSTAR additionally models temporal dependencies through transformer blocks. Compared with DSRNN, which relies on recurrent node and edge RNNs, NaviSTAR uses self-attention and graph convolution to represent crowd dynamics. This enables a fair comparison between recur-

rent, attention-based, graph-based, and transformer-based social navigation policies under the same PPO training pipeline, reward function, action space, and evaluation protocol.

#### B.4.9 NAVIDIFF WANG ET AL. (2025B)

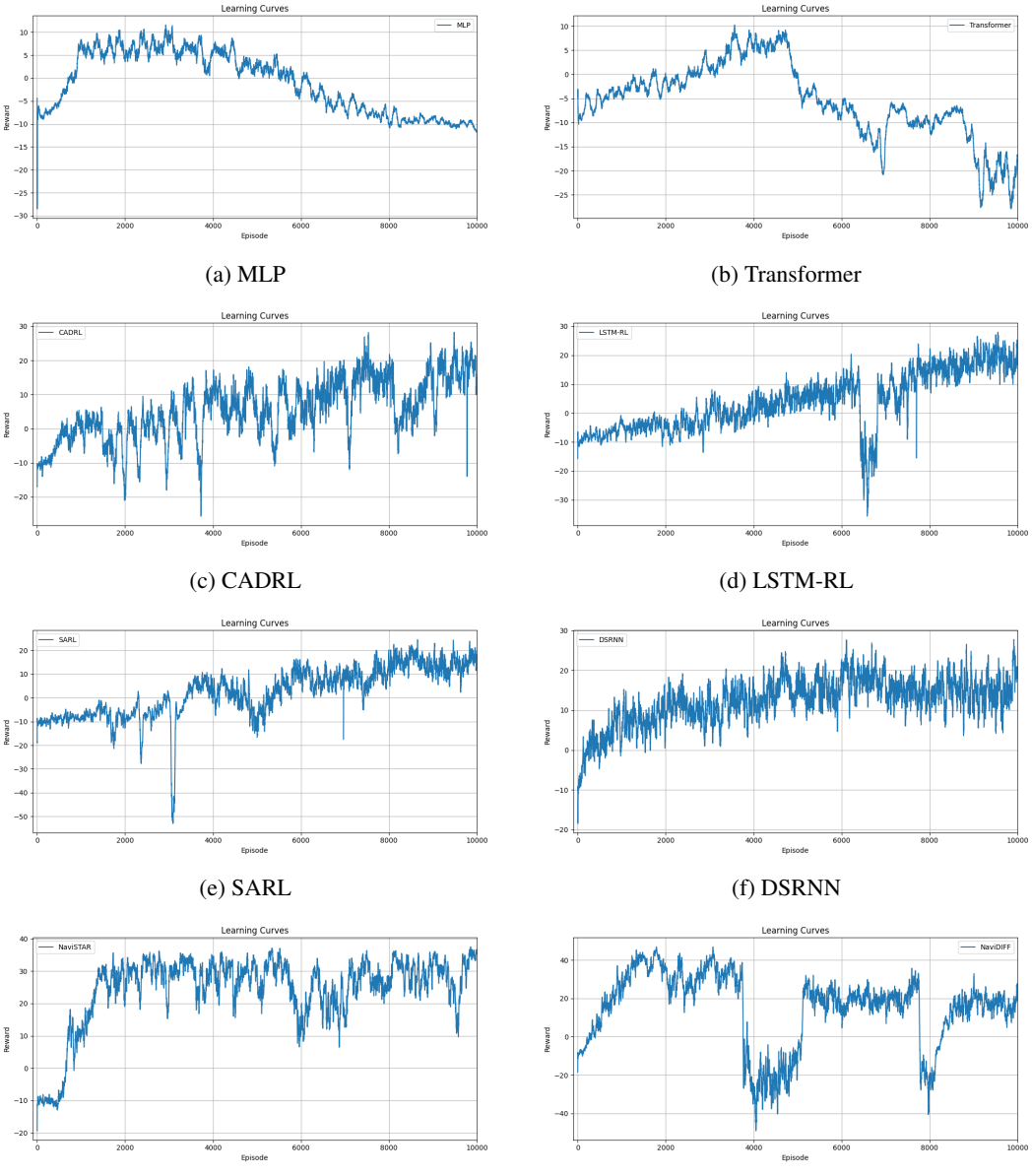
NaviDIFF is a Hamiltonian-constrained socially-aware navigation framework that models human-robot interaction from both physical and social perspectives. Different from standard learning-based navigation policies that directly map observations to robot actions, NaviDIFF explicitly introduces a port-Hamiltonian formulation to describe the physical interaction dynamics among the robot and pedestrians. This provides an energy-based representation of social navigation, where human-robot interactions are modeled through physically-informed terms such as system energy, interaction energy, and dissipative effects.

In NaviDIFF, the surrounding humans and the robot are represented as a spatial-temporal interaction system. A spatial-temporal transformer is used to capture both social dependencies among agents and temporal dependencies across navigation history. Based on these encoded interaction features, NaviDIFF constructs Hamiltonian-constrained social navigation features that describe the closed-loop physical interaction process between the robot and pedestrians.

To further handle the uncertainty of human-robot cooperation, NaviDIFF introduces a diffusion network to approximate the cooperative and competitive interaction distribution. Instead of treating pedestrian interactions as deterministic, NaviDIFF uses the diffusion module to model uncertain social cooperation patterns among agents. This helps the robot reason about possible interaction outcomes in dense and dynamic crowds.

In our benchmark, NaviDIFF is used as a learning-based baseline for socially-aware robot navigation. It receives the robot state, goal information, temporal motion features, and surrounding human interaction features. The Hamiltonian-constrained and diffusion-enhanced representation is then used by the policy to generate the robot navigation command. Compared with NaviSTAR, NaviDIFF further introduces port-Hamiltonian interaction modeling and diffusion-based cooperative distribution estimation. Compared with DSRNN, SARL, and LSTM-RL, NaviDIFF does not only encode human-robot interactions through recurrence or attention, but also incorporates physics-informed interaction constraints and uncertainty modeling. Therefore, NaviDIFF serves as a strong Hamiltonian-constrained and diffusion-enhanced baseline in our social navigation benchmark.

### B.5 LEARNING CURVE




(a) MLP (b) Transformer

(c) CADRL (d) LSTM-RL

(e) SARL (f) DSRNN

(g) NaviSTAR (h) NaviDIFF

Figure 34: Training reward curves of different navigation baselines, including MLP, Transformer, CADRL, LSTM-RL, SARL, DSRNN, NaviSTAR, and NaviDIFF.

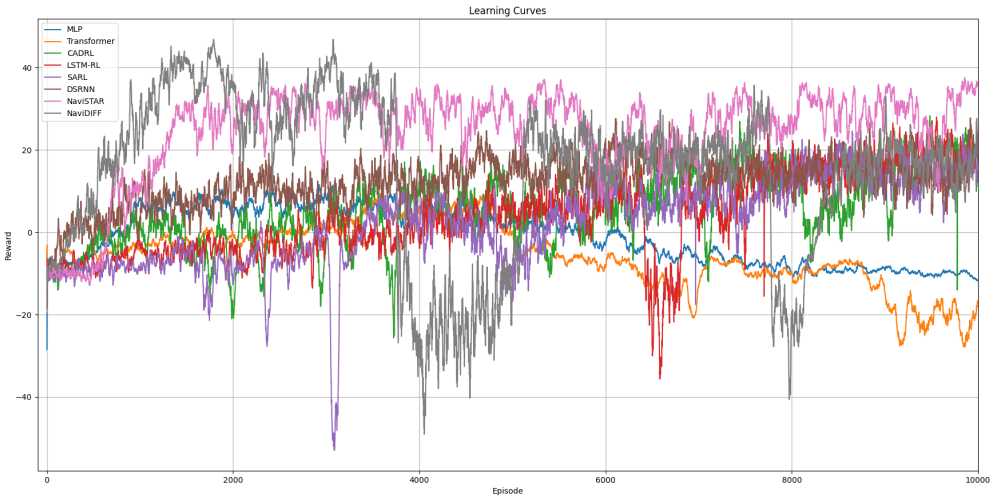



Figure 35: Training reward curves of different navigation baselines.

### B.6 Evaluation Metrics

To comprehensively evaluate the robot's social navigation performance, we consider four major aspects: task completion, safety, social comfort, and navigation efficiency. Specifically, the evaluation includes success rate, collision rate, uncomfortable interaction rate, navigation time, navigation path length, minimum human distance, and a unified social score. Among these metrics, success rate measures whether the robot can reach the target, collision rate reflects navigation safety, uncomfortable interaction rate evaluates social compliance, and navigation time and path length measure motion efficiency. The social score further combines these factors into a single normalized score for overall comparison.

**Success Rate.** The success rate measures the proportion of evaluation episodes in which the robot successfully reaches the target goal. An episode is regarded as successful if the robot-goal distance is smaller than the predefined goal-reaching threshold. Given $N_{\text{eval}}$ evaluation episodes and a binary success indicator $\mathbb{I}_i^{\text{succ}}$ for the $i$-th episode, the success rate is defined as

$$\text{SR} = \frac{1}{N_{\text{eval}}} \sum_{i=1}^{N_{\text{eval}}} \mathbf{1}_i^{\text{succ}}, \tag{36}$$

where

$$\mathbf{1}_i^{\text{succ}} = \begin{cases} 1, & d_{g,i}^{\text{final}} < d_{\text{goal}}, \\ 0, & \text{otherwise.} \end{cases} \tag{37}$$

Here, $d_{g,i}^{\text{final}}$ is the final distance between the robot and the target in episode $i$, and $d_{\text{goal}}$ is the goal-reaching threshold.

**Navigation Path Length.** The navigation path length measures the actual distance traveled by the robot during an episode. It is computed by accumulating the displacement of the robot between consecutive simulation steps. For the $i$-th episode, the path length is defined as

$$L_i = \sum_{t=1}^{T_i} \left\| \mathbf{p}_{i,t}^r - \mathbf{p}_{i,t-1}^r \right\|_2, \tag{38}$$

where $\mathbf{p}_{i,t}^r$ denotes the 2D position of the robot at timestep $t$, and $T_i$ is the number of timesteps in episode $i$. The average navigation path length over all finished evaluation episodes is

$$\bar{L} = \frac{1}{N_{\text{fin}}} \sum_{i=1}^{N_{\text{fin}}} L_i, \tag{39}$$

where $N_{\text{fin}}$ is the number of finished evaluation episodes.

**Navigation Time.** The navigation time measures the duration required by the robot to finish an episode. It is computed from the number of executed simulation steps and the environment timestep. For the $i$-th episode, the navigation time is given by

$$T_i^{\text{nav}} = n_i \Delta t, \tag{40}$$

where $n_i$ is the number of executed environment steps and $\Delta t$ is the simulation timestep. The average navigation time is computed as

$$\bar{T}^{\text{nav}} = \frac{1}{N_{\text{fin}}} \sum_{i=1}^{N_{\text{fin}}} T_i^{\text{nav}}. \tag{41}$$

**Social Score.** To provide an overall measurement of social navigation performance, we define a unified social score that jointly considers success, safety, comfort, and efficiency. The success score is directly given by the success rate, while the safety score penalizes collisions and the comfort score penalizes uncomfortable interactions:

$$S_{\text{succ}} = \text{SR}, \quad S_{\text{safety}} = 1 - R_{\text{col}}, \quad S_{\text{comfort}} = 1 - R_{\text{unc}}, \tag{42}$$

where $R_{\text{col}}$ is the collision rate and $R_{\text{unc}}$ is the average uncomfortable rate. The efficiency score combines both navigation time and path length:

$$S_{\text{time}} = 1 - \frac{\bar{T}^{\text{nav}}}{T_{\max}}, \quad S_{\text{path}} = \frac{R_g}{\bar{L}}, \tag{43}$$

where $T_{\max}$ is the maximum episode duration and $R_g$ is the goal radius. The final efficiency score is

$$S_{\text{eff}} = \frac{1}{2} S_{\text{time}} + \frac{1}{2} S_{\text{path}}. \tag{44}$$

The final social score is computed as

$$S_{\text{social}} = 100 \left( k_1 S_{\text{succ}} + k_2 S_{\text{safety}} + k_3 S_{\text{comfort}} + k_4 S_{\text{eff}} \right). \tag{45}$$

In implementation, each normalized sub-score is clipped to the range $[0, 1]$ to avoid abnormal values caused by extreme episode outcomes.

**Other Metrics.** The collision rate measures the proportion of episodes in which the robot collides with pedestrians. The timeout rate measures the proportion of episodes that reach the maximum episode length without success. The died rate measures the proportion of episodes where the robot falls or produces invalid states. The uncomfortable interaction rate measures how often the robot enters the predefined social discomfort region around pedestrians. The average minimum human distance records the closest robot-human distance during each episode and reflects the safety margin. The average return and average reward per step are mainly used to analyze training quality and reward optimization behavior.

Table 4: Main PPO Hyperparameters Table.

| Parameter | Value |
|---|---|
| Discount factor | 0.99 |
| GAE parameter | 0.95 |
| PPO clipping range | 0.2 |
| Learning rate | $4 \times 10^{-5}$ |
| Rollout length | 24 |
| Mini-batch size | 4 |
| Optimization epochs | 5 |
| Entropy coefficient | 0.005 |
| Value loss coefficient | 0.5 |
| Maximum gradient norm | 1.0 |
| Target KL divergence | 0.01 |
| Advantage normalization | True |
| Action distribution | Gaussian |
| Initial log standard deviation | 0.0 |
| Log standard deviation range | $[-20, 2]$ |

### B.7 PPO CONFIGURATION

In the benchmark, we uniformly use Proximal Policy Optimization (PPO) to train the navigation policies, as shown in Table. 4. PPO follows an actor-critic framework, where the actor predicts the robot action distribution and the critic estimates the state value. Given the observation $o_t$, the actor outputs a Gaussian policy for the continuous velocity action $a_t$, while the critic predicts the value function $V_\phi(o_t)$. PPO improves training stability by limiting the policy update through a clipped surrogate objective.

The actor loss is defined as the clipped PPO objective:

$$\mathcal{L}_{\text{actor}} = -\mathbb{E}_t \left[ \min \left( \frac{\pi_\theta(a_t \mid o_t)}{\pi_{\theta_{\text{old}}}(a_t \mid o_t)} \hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}_t \right) \right], \tag{46}$$

where $\hat{A}_t$ is the estimated advantage and $\epsilon$ is the PPO clipping range.

The critic loss is used to optimize the value function:

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_t \left[ \left( V_\phi(o_t) - \hat{R}_t \right)^2 \right], \tag{47}$$

where $\hat{R}_t$ is the target return.

The final PPO loss combines the actor loss, critic loss, and entropy regularization:

$$\mathcal{L}_{\text{PPO}} = \mathcal{L}_{\text{actor}} + c_v \mathcal{L}_{\text{critic}} - c_{\text{ent}} \mathbb{E}_t \left[ \mathcal{H}\left(\pi_\theta(\cdot \mid o_t)\right) \right], \tag{48}$$

where $c_v$ is the value loss coefficient, $c_{\text{ent}}$ is the entropy coefficient, and $\mathcal{H}(\cdot)$ denotes the policy entropy.

The reward function trains the high-level path planner to output socially compliant velocity commands $a_t = [v_x, v_y, \omega_z]$. It encourages the robot to move toward the goal, reduce the goal distance, reach the target, avoid human collisions, and maintain a comfortable distance from pedestrians. The total reward is defined as

$$\begin{aligned} r_t = {} & 1\, r_t^{\text{dir}} \Delta t + 2\, r_t^{\text{prog}} - 1\, r_t^{\text{dist}} \Delta t + 10\, r_t^{\text{goal}} \\ & - 20\, r_t^{\text{col}} - 0.5\, r_t^{\text{soc}} \Delta t. \end{aligned} \tag{49}$$

The goal-direction reward is computed by the cosine alignment between the commanded planar velocity and the goal direction:

$$r_t^{\text{dir}} = \frac{1}{2} \left( \frac{\mathbf{v}_t^{\text{cmd}}}{\|\mathbf{v}_t^{\text{cmd}}\|_2} \cdot \frac{\mathbf{g}_t}{\|\mathbf{g}_t\|_2} + 1 \right), \tag{50}$$

where $\mathbf{v}_t^{\mathrm{cmd}} = [v_x, v_y]$ and $\mathbf{g}_t$ is the goal vector in the robot body frame. This term is set to zero when the commanded speed is smaller than $0.05$.

The progress reward is

$$r_t^{\mathrm{prog}} = d_{t-1}^g - d_t^g, \tag{51}$$

where $d_t^g$ is the current robot-goal distance. The goal-distance penalty is

$$r_t^{\mathrm{dist}} = d_t^g. \tag{52}$$

The goal-reaching reward is

$$r_t^{\mathrm{goal}} = \mathbf{1}\left(d_t^g < d_{\mathrm{goal}}\right). \tag{53}$$

For social safety, the collision penalty is

$$r_t^{\mathrm{col}} = \mathbf{1}\left(\min_i d_{r,h_i,t} < r_r + r_h\right), \tag{54}$$

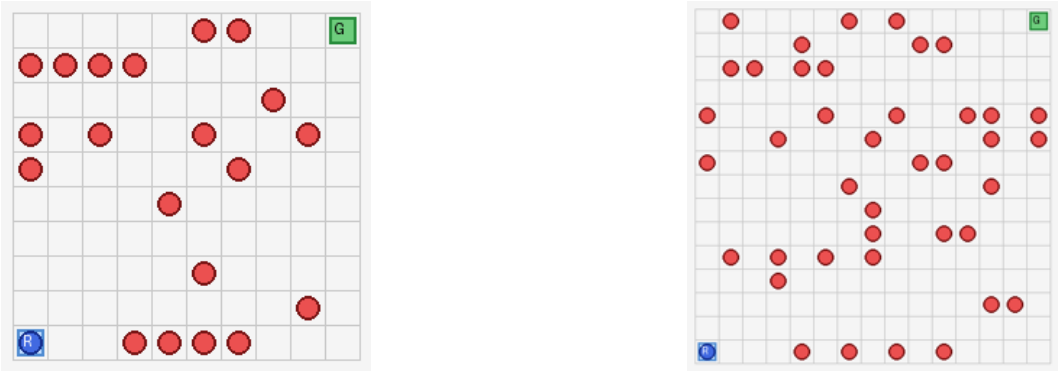
where $r_r = 0.65\ \mathrm{m}$ and $r_h = 0.30\ \mathrm{m}$. Thus, the collision distance is $0.95\ \mathrm{m}$.

The social-distance penalty is

$$r_t^{\mathrm{soc}} = \sum_i \max\left(0, d_{\mathrm{social}} - d_{r,h_i,t}\right), \tag{55}$$

where the social distance $d_{\mathrm{comfort}}$ is supposed as 0.45 m Rios-Martinez et al. (2015)

$$d_{\mathrm{social}} = r_r + r_h + d_{\mathrm{comfort}} = 0.65 + 0.30 + 0.45 = 1.4\ \mathrm{m}. \tag{56}$$



(a) Simple-level scenario.

(b) Hard-level scenario.

Figure 36: Visualization of grid-world environments with distinct difficulty levels.

# C THE GRID-BASED DISCRETE SOCIAL NAVIGATION EXPERIMENTS

## C.1 TASK CONFIGURATION

To conduct the ablation studies, we construct a simplified grid-world social navigation environment to evaluate the RL-based fine-tuning performance of distinct variants of the proposed NaviVLA framework. As shown in Fig.36, the environment contains a mobile robot, a navigation destination, and multiple dynamic obstacles that move between neighboring free cells. NaviVLA is adapted to this grid-world setting, where, at each decision step, the robot controller generates a discrete navigation action ($a \in \mathcal{A}_{gw} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$) to interact with the environment. Moreover, the handcrafted reward function $\mathcal{R}_{gw}$ encourages the robot to reach the goal efficiently while penalizing invalid movements and unnecessary delays, collisions with dynamic obstacles, and failure to reach the destination within the predefined episode horizon.

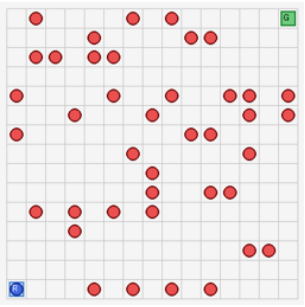
As shown in Fig. 36a, the simple-level scenario contains 20 dynamic obstacles in a $10 \times 10$ grid. In contrast, the difficult-level scenario contains 40 dynamic obstacles in a $15 \times 15$ grid, as illustrated in Fig. 36b. In both scenarios, the robot and its goal are initialized at fixed, diagonally opposite corners of the grid, while the dynamic obstacles are randomly distributed throughout the environment.

The VLA input consists of a rendered grid image, a textual description of the robot observation, as well as the robot observation vector. The visual and textual input are processed by the VLM to extract a multimodal hidden feature representation, which is subsequently decoded together with robot state by the action expert to generate the navigation policy. Then the VLA model is fine-tuned by PPO algorithm. As illustrated in Fig. 37 and Fig. 38, the input image provides a visual representation of the grid-world, including the robot, goal, and dynamic obstacles, while the corresponding text prompt, converted from the robot's observation state, describes the environment configuration, robot and goal positions, obstacle positions and velocities, available actions, and navigation objective. Together, these multimodal observations are provided to the VLA policy to construct a hidden spatial representation for safe next-action selection.

The overall VLA frameowkr remains consistent with NaviVLA, while the input dimensions, prompt design, and network scale are adapted to the simplified grid-world environment, as shown in Table. 5.

Table 5: Comparison of grid-world and IsaacSim social navigation environments.

| **Component** | **Grid-World Environment** | **IsaacSim Environment** |
|---|---|---|
| Image Input | Single grid image | Multi-camera and BEV images |
| Text Input | Env prompt + direct state-to-text conversion obs prompt | Env prompt + world model state-to-text conversion obs prompt |
| State Input | Discrete observation | Continuous observation |
| Human | Randomly moving dynamic grid cells | Humans controlled by ORCA, Social Force, or diffusion model |
| Robot | Grid-cell robot agent | Quadruped robot with the locomotion policy |
| Environment | Fixed size two-dimensional grid world environment | Physics-based three-dimensional IsaacSim environment |
| Actor | Four discrete action logits | Continuous action distribution parameterized by $\{\mu_h(\theta),\ \sigma_h(\theta)\}$ |
| Critic | State-value estimator | State-value estimator |
| Action Output | Discrete {`up`, `down`, `left`, or `right`} | Continuous $[v_x, v_y, \omega_z]$ |

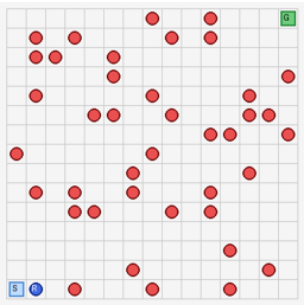


*The image input to the VLA policy.*

```
<|im_start|>user
<|vision_start|><|image_pad|><|vision_end|>

Environment Prompt:

You control a robot navigating a dynamic grid-world. The objective is to reach the goal efficiently while
      remaining inside the grid and avoiding collisions with dynamic obstacles. The environment uses (x,
      y) coordinates. Each obstacle is described by its current position and velocity, where velocity (
     vx, vy) indicates its most recent movement direction. Available actions are: up: (x, y + 1); down:
     (x, y - 1); left: (x - 1, y); right: (x + 1, y). Use the robot state, obstacle states, and visual
     observation together to construct a spatial representation of the environment for safe next action
     selection.


Robot Observation Prompt:

Grid size: 15x15.

Robot position: (0, 0).

Goal position: (14, 14).

Dynamic obstacles: obs0: pos=(12, 9), vel=(0, 0); obs1: pos=(7, 4), vel=(0, 0); obs2: pos=(4, 0), vel=(0,
      0); obs3: pos=(2, 12), vel=(0, 0); obs4: pos=(9, 13), vel=(0, 0); obs5: pos=(7, 9), vel=(0, 0);
     obs6: pos=(14, 10), vel=(0, 0); obs7: pos=(9, 8), vel=(0, 0); obs8: pos=(8, 14), vel=(0, 0); obs9:
     pos=(4, 12), vel=(0, 0); obs10: pos=(10, 0), vel=(0, 0); obs11: pos=(5, 12), vel=(0, 0); obs12: pos
     =(8, 0), vel=(0, 0); obs13: pos=(11, 10), vel=(0, 0); obs14: pos=(12, 2), vel=(0, 0); obs15: pos
     =(1, 12), vel=(0, 0); obs16: pos=(0, 8), vel=(0, 0); obs17: pos=(1, 4), vel=(0, 0); obs18: pos=(7,
     6), vel=(0, 0); obs19: pos=(6, 0), vel=(0, 0); obs20: pos=(0, 10), vel=(0, 0); obs21: pos=(3, 9),
     vel=(0, 0); obs22: pos=(11, 5), vel=(0, 0); obs23: pos=(6, 14), vel=(0, 0); obs24: pos=(5, 10), vel
     =(0, 0); obs25: pos=(14, 9), vel=(0, 0); obs26: pos=(12, 10), vel=(0, 0); obs27: pos=(10, 5), vel
     =(0, 0); obs28: pos=(13, 2), vel=(0, 0); obs29: pos=(8, 10), vel=(0, 0); obs30: pos=(12, 7), vel
     =(0, 0); obs31: pos=(5, 4), vel=(0, 0); obs32: pos=(6, 7), vel=(0, 0); obs33: pos=(10, 13), vel=(0,
      0); obs34: pos=(1, 14), vel=(0, 0); obs35: pos=(7, 5), vel=(0, 0); obs36: pos=(10, 8), vel=(0, 0);
      obs37: pos=(3, 4), vel=(0, 0); obs38: pos=(4, 13), vel=(0, 0); obs39: pos=(3, 3), vel=(0, 0).


Image Observation Prompt:
The image shows the grid map: blue is the robot, green is the goal, red circles are dynamic obstacles,
     and S is the start. Available actions: up, down, left, right. Return a hidden navigation
     representation for choosing the next action.


<|im_end|>
```

*The text input to the VLA policy.*

Figure 37: Example multimodal input for the grid-based VLA navigation policy in step-1.



*The image input to the VLA policy.*

```
<|im_start|>user
<|vision_start|><|image_pad|><|vision_end|>

Environment Prompt:

You control a robot navigating a dynamic grid-world. The objective is to reach the goal efficiently while
      remaining inside the grid and avoiding collisions with dynamic obstacles. The environment uses (x,
      y) coordinates. Each obstacle is described by its current position and velocity, where velocity (
     vx, vy) indicates its most recent movement direction. Available actions are: up: (x, y + 1); down:
     (x, y - 1); left: (x - 1, y); right: (x + 1, y). Use the robot state, obstacle states, and visual
     observation together to construct a spatial representation of the environment for safe next action
     selection.


Robot Observation Prompt:

Grid size: 15x15.

Robot position: (1, 0).

Goal position: (14, 14).

Dynamic obstacles: obs0: pos=(13, 9), vel=(1, 0); obs1: pos=(8, 4), vel=(1, 0); obs2: pos=(3, 0), vel
     =(-1, 0); obs3: pos=(1, 12), vel=(-1, 0); obs4: pos=(10, 13), vel=(1, 0); obs5: pos=(7, 10), vel
     =(0, 1); obs6: pos=(14, 11), vel=(0, 1); obs7: pos=(10, 8), vel=(1, 0); obs8: pos=(8, 13), vel=(0,
     -1); obs9: pos=(5, 12), vel=(1, 0); obs10: pos=(11, 0), vel=(1, 0); obs11: pos=(5, 11), vel=(0, -1)
     ; obs12: pos=(7, 0), vel=(-1, 0); obs13: pos=(12, 10), vel=(1, 0); obs14: pos=(11, 2), vel=(-1, 0);
      obs15: pos=(2, 12), vel=(1, 0); obs16: pos=(0, 7), vel=(0, -1); obs17: pos=(1, 5), vel=(0, 1);
     obs18: pos=(7, 7), vel=(0, 1); obs19: pos=(6, 1), vel=(0, 1); obs20: pos=(1, 10), vel=(1, 0); obs21
     : pos=(4, 9), vel=(1, 0); obs22: pos=(10, 5), vel=(-1, 0); obs23: pos=(7, 14), vel=(1, 0); obs24:
     pos=(5, 9), vel=(0, -1); obs25: pos=(14, 8), vel=(0, -1); obs26: pos=(12, 9), vel=(0, -1); obs27:
     pos=(10, 4), vel=(0, -1); obs28: pos=(13, 1), vel=(0, -1); obs29: pos=(8, 9), vel=(0, -1); obs30:
     pos=(12, 6), vel=(0, -1); obs31: pos=(4, 4), vel=(-1, 0); obs32: pos=(6, 6), vel=(0, -1); obs33:
     pos=(10, 14), vel=(0, 1); obs34: pos=(1, 13), vel=(0, -1); obs35: pos=(6, 5), vel=(-1, 0); obs36:
     pos=(11, 8), vel=(1, 0); obs37: pos=(3, 5), vel=(0, 1); obs38: pos=(3, 13), vel=(-1, 0); obs39: pos
     =(3, 4), vel=(0, 1).


Image Observation Prompt:
The image shows the grid map: blue is the robot, green is the goal, red circles are dynamic obstacles,
     and S is the start. Available actions: up, down, left, right. Return a hidden navigation
     representation for choosing the next action.


<|im_end|>
```

*The text input to the VLA policy.*

Figure 38: Example multimodal input for the grid-based VLA navigation policy in step-2.

### C.2 RL Fine-Tuning Procedure for the VLA Policy

Unlike the conventional RL-based agent, the RL-VLA fine-tuning framework preserves not only standard PPO quantities, such as actions, rewards, log-probabilities, value estimates, and advantages, but also the corresponding image and text observations required to reconstruct the VLA policy distribution during optimization. In addition, the actor adopts different output formulations across the two environments, producing discrete actions in the grid-world environment while modeling a continuous action distribution in IsaacSim. Thus, RL-VLA retains the standard PPO optimization mechanism while extending the observation representation, databuffer structure, and policy architecture to support multimodal VLA-based fine-tuning procedure.

Herein, we employ the PPO algorithm Schulman et al. (2017) in both the discrete grid-world navigation environment and the continuous IsaacSim navigation environment. The databuffer collects the action log-probabilities and value predictions generated by the old policy, together with the multimodal observations, executed actions, rewards, and termination signals. After collecting fixed-length rollouts from multiple parallel environments, these transitions are used to compute returns and generalized advantage estimates. The resulting mini-batches are then repeatedly sampled from the buffer to update the VLA actor and critic using the PPO objective in Equation-2.

Despite the grid-world navigation simulation environment is intentionally simplified, we retain the main VLA architecture from NaviVLA and the proposed RL-VLA policy learning mechanism to study the effectiveness of distinct fine-tuning strategies. Specifically, we evaluate three training paradigms for three VLMs (Qwen3.5-0.8B, Qwen3.5-0.8B-base, and Qwen3.5-9B), involving a frozen-VLM setting in which only the action expert and actor-critic policy are optimized, a LoRA-based VLA setting in which lightweight adapters are jointly fine-tuned with the policy, and a frozen-to-LoRA setting in which training begins with a frozen VLM and subsequently transitions to LoRA fine-tuning. Moreover, the proposed RL-VLA framework incorporates multimodal perception, VLA policy learning, and closed-loop interaction for social navigation. At each timestep, the VLA network receives visual images, structured text prompts, and robot observation state. These multimodal inputs are encoded into a VLM's hidden feature representation, which is subsequently processed by the actor and critic blocks for RL optimization. For all ablation models, we use the same PPO learning framework and optimization settings, while varying the databuffer parameters, including the batch size, minibatch size, and number of parallel environments. Eventually, the Qwen3.5-0.8B and Qwen3.5-0.8B-Base experiments are conducted in a $10 \times 10$ cell grid-world containing 20 randomly moving humans, with each episode limited to 100 steps. The Qwen3.5-9B experiments are conducted in a more challenging $15 \times 15$ grid-world containing 40 randomly moving humans, with each episode limited to 200 steps. Evaluation metrics are computed over 50 randomly generated validation cases. SR denotes the success rate, while Avg. Time denotes the average number of steps across all evaluated trajectories. Update represents the number of PPO optimization updates, Para. denotes the order of trainable parameter complexity, and Init. specifies the model or checkpoint used for initialization.

Initial-1 uses Qwen3.5-0.8B with text only input and is evaluated before PPO training, while Initial-2 uses the same backbone with multimodal text and image multi input. Frozen-1 freezes Qwen3.5-0.8B and trains other components using text input with a batch size of 128, a minibatch size of 16, and one parallel environment. Frozen-2 uses the same frozen paradigm with a batch size of 1024, a minibatch size of 128, and 128 parallel environments. Frozen-3 follows the same large-scale configuration as Frozen-2 but replaces the VLM backbone with Qwen3.5-0.8B-Base. Frozen-4 uses frozen Qwen3.5-0.8B with multimodal input, a batch size of 1024, a minibatch size of 128, and 64 parallel environments. Frozen-to-LoRA-0.8B is initialized from the Frozen-1 checkpoint with an SR of $54\%$ and then fine-tuned using LoRA with text input, a rollout batch size of 128, a minibatch size of 8, and one parallel environment. LoRA directly applies LoRA fine-tuning to initial-1 using the same databuffer configuration with batch number of 128, a minibatch size of 8, and single parallel environment.

Initial-3 uses Qwen3.5-9B with multimodal input and is evaluated before PPO optimization. Frozen-5 freezes the Qwen3.5-9B backbone and trains other modules with a batch size of 1024, a minibatch size of 64, and 100 parallel environments. Frozen-6 uses the same Qwen3.5-9B architecture and databuffer configuration as Frozen-5 but is evaluated after more PPO updates. Frozen-to-LoRA-9B is initialized from the Frozen-5 checkpoint with an SR of $64\%$ and applies LoRA fine-tuning with a batch size of 128, a minibatch size of 16, and two parallel environments.

### C.3 RL-VLA Fine-Tuning Procedure Experimental nalysis

As shown in Table 6, all initial policies achieve a $0\%$ success rate without PPO training because the randomly initialized action expert and actor cannot accurately decode the VLM outputs. For the 0.8B models, Frozen-2 obtains the best performance with a $100\%$ success rate and an average navigation time of 19.37 steps, while Frozen-3 achieves a comparable $98\%$ success rate. The multimodal Frozen-4 model produces the shortest average navigation time of 18.76 steps but a lower success rate of $84\%$. Both LoRA-based models improve over their initial policies, and initialization from Frozen-1 performs slightly better than direct LoRA training. For the 9B models, Frozen-6 and

Frozen-to-LoRA both achieve a $98\%$ success rate, showing that PPO training effectively improves the larger multimodal model.

To systematically analyze the effectiveness and training characteristics of Discrete-NaviVLA, we formulate the following three main research questions for this ablation study.

**Question 1: Can a VLA model be employed for social navigation?**

**Answer:** Yes. As shown in Table 6, both the 0.8B and 9B Discrete-NaviVLA models can learn effective navigation policies after PPO optimization. Although all initial policies obtain a $0\%$ success rate, the trained models achieve success rates ranging from $54\%$ to $100\%$. In particular, Frozen-2 reaches a $100\%$ success rate in the $10 \times 10$ environment, while Frozen-6 reaches a $98\%$ success rate in the more challenging $15 \times 15$ environment. These results demonstrate that a VLA architecture can be effectively adapted to discrete social navigation.

**Question 2: Can RL effectively fine-tune a VLA model for navigation tasks?**

**Answer:** Yes. PPO effectively improves Discrete-NaviVLA under both frozen-backbone and LoRA-based training strategies. The frozen 0.8B models improve from an initial success rate of $0\%$ to as high as $100\%$, while the frozen 9B model improves from $0\%$ to $98\%$. LoRA fine-tuning further improves the 0.8B model from $54\%$ to $78\%$ and the 9B model from $64\%$ to $98\%$.

Although both frozen-backbone and LoRA-based training demonstrate strong performance in this simplified discrete navigation task, LoRA has a higher representational capacity because it introduces substantially more trainable parameters. Therefore, LoRA may provide greater potential for learning more complex navigation behaviors in challenging environments, although this advantage is not fully reflected in the current discrete benchmark.

**Question 3: Which training parameters have a significant effect on RL-VLA performance?**

**Answer:** The batch size, minibatch size, amount of collected rollout data, and number of parallel environments have substantial effects on RL-VLA training performance. For example, Frozen-1 uses a batch size of 128, a minibatch size of 16, and one parallel environment, and achieves a $54\%$ success rate after 1000 PPO updates. In contrast, Frozen-2 uses a rollout batch size of 1024, a minibatch size of 128, and 128 parallel environments, and reaches a $100\%$ success rate after only 60 updates. Since PPO is an on-policy reinforcement learning algorithm, the scale, diversity, and quality of the collected databuffer are critical to stable policy optimization and sample-efficient training. A larger databuffer and more parallel environments can provide more diverse interaction trajectories and improve the probability of collecting expert trajecotries.

Additionally, multimodal input appears to require more optimization than text-only input under the current configurations. For example, the text-only Frozen-2 model reaches a $100\%$ success rate after 60 updates, whereas the multimodal Frozen-4 model achieves $84\%$ under the same number of updates. This suggests that jointly learning from visual and textual observations may introduce additional optimization difficulty. Finally, Qwen3.5-0.8B and Qwen3.5-0.8B-Base show no substantial performance difference in this task, as Frozen-2 and Frozen-3 achieve success rates of $100\%$ and $98\%$, respectively.

### C.4 RLHF Fine-Tuning Procedure for the VLA Policy

To systematically analyze the effectiveness and training characteristics of preference learning for NaviVLA, we formulate the following three main research questions for this ablation study.

**Question 1: Can preference learning teach VLA navigation behaviors that are not explicitly encoded by the original reward function?**

**Answer:** Yes. As shown in Table 7, trajectory-level DPO can effectively modify navigation behavior according to different preference objectives while largely preserving the original task capability. For example, under Qwen3.5-0.8B frozen tuning, Long-Time DPO increases the chosen trajectory metric from approximately 19 to 27.28 at update 60 while maintaining a $96\%$ evaluation success rate. Similarly, Long-Path DPO increases the chosen metric to $24.08$ while retaining a $90\%$ success rate. Short-Time and Short-Path preferences can also modify the corresponding trajectory characteristics without substantially degrading navigation success. These results demonstrate that preference learn-

Table 6: Experimental configurations and performance of Discrete-NaviVLA.

| Ablation Model | VLM | Input | Update | Para. | Init. | SR | Avg. Time |
|---|---|---|---|---|---|---|---|
| Initial-1 | Qwen3.5-0.8B | Text | 0 | – | – | 0% | 100 |
| Initial-2 | Qwen3.5-0.8B | Multi | 0 | – | – | 0% | 100 |
| Frozen-1 | Qwen3.5-0.8B | Text | 1000 | $\mathcal{O}(10^5)$ | Initial-1 | 54% | 20.78 |
| Frozen-2 | Qwen3.5-0.8B | Text | 60 | $\mathcal{O}(10^5)$ | Initial-1 | 100% | 19.37 |
| Frozen-3 | Qwen3.5-0.8B-Base | Text | 60 | $\mathcal{O}(10^5)$ | Initial-1 | 98% | 19.48 |
| Frozen-4 | Qwen3.5-0.8B | Multi | 60 | $\mathcal{O}(10^5)$ | Initial-2 | 84% | 18.76 |
| Frozen-to-LoRA | Qwen3.5-0.8B | Text | 1000 | $\mathcal{O}(10^6)$ | Frozen-1 | 78% | 19.46 |
| LoRA | Qwen3.5-0.8B | Text | 1000 | $\mathcal{O}(10^6)$ | Initial-1 | 75% | 26.93 |
| Initial-3 | Qwen3.5-9B | Multi | 0 | – | – | 0% | 200 |
| Frozen-5 | Qwen3.5-9B | Multi | 15 | $\mathcal{O}(10^6)$ | Initial-3 | 64% | 28.63 |
| Frozen-6 | Qwen3.5-9B | Multi | 45 | $\mathcal{O}(10^6)$ | Initial-3 | 98% | 29.92 |
| Frozen-to-LoRA | Qwen3.5-9B | Multi | 1000 | $\mathcal{O}(10^7)$ | Frozen-5 | 98% | 31.02 |

Table 7: Preference learning dynamics of NaviVLA under different DPO objectives and model configurations.

| VLM / Tuning | Preference Target | Pairs | Update | Pref. Acc. | Reward Δ | Chosen Metric | Rejected Metric | Eval. SR |
|---|---|---|---|---|---|---|---|---|
| Qwen3.5-0.8B Freeze | Short-Time DPO | – | 0 | – | – | – | – | 92% |
| | | 128 | 15 | 0.602 | 0.0140 | 18.86 | 18.84 | 94% |
| | | 128 | 30 | 0.598 | 0.0121 | 19.02 | 19.11 | 88% |
| | | 128 | 45 | 0.492 | 0.0217 | 18.98 | 19.07 | 98% |
| | | 128 | 60 | 0.574 | 0.0246 | 18.87 | 18.92 | 96% |
| | Short-Path DPO | – | 0 | – | – | – | – | 92% |
| | | 128 | 15 | 0.539 | 0.0274 | 18.48 | 17.83 | 92% |
| | | 128 | 30 | 0.570 | 0.0491 | 18.69 | 17.61 | 92% |
| | | 128 | 45 | 0.605 | 0.0539 | 18.28 | 17.92 | 98% |
| | | 128 | 60 | 0.602 | 0.0623 | 18.41 | 17.65 | 96% |
| | Long-Time DPO | – | 0 | – | – | – | – | 80% |
| | | 32 | 15 | 0.812 | 0.4621 | 27.69 | 17.16 | 96% |
| | | 32 | 30 | 0.688 | 0.3559 | 26.63 | 18.63 | 90% |
| | | 32 | 45 | 0.594 | 0.4485 | 27.91 | 18.81 | 98% |
| | | 32 | 60 | 0.719 | 0.4409 | 27.28 | 19.75 | 96% |
| | Long-Path DPO | – | 0 | – | – | – | – | 92% |
| | | 128 | 15 | 0.684 | 0.2546 | 22.22 | 16.66 | 94% |
| | | 128 | 30 | 0.781 | 0.3800 | 22.89 | 17.65 | 88% |
| | | 128 | 45 | 0.727 | 0.3621 | 22.77 | 16.97 | 94% |
| | | 128 | 60 | 0.805 | 0.5411 | 24.08 | 17.27 | 90% |
| Qwen3.5-9B LoRA | Long-Path DPO | – | 0 | – | – | – | – | 100% |
| | | 8 | 15 | 0.250 | -0.0013 | 31.75 | 26.50 | 98% |
| | | 8 | 30 | 0.312 | 0.0004 | 30.50 | 27.13 | 94% |
| | | 8 | 45 | 0.625 | -0.0005 | 61.25 | 40.50 | 78% |
| | | 8 | 60 | 0.250 | -0.0347 | 48.50 | 37.75 | 82% |
| | | 8 | 100 | 0.750 | -0.0002 | 48.75 | 44.00 | 100% |
| | Long-Time DPO | – | 0 | – | – | – | – | 100% |
| | | 8 | 15 | 0.375 | -0.0007 | 31.88 | 29.00 | 98% |
| | | 8 | 30 | 0.875 | 0.0364 | 58.38 | 40.00 | 60% |
| | | 8 | 45 | 0.625 | 0.0219 | 146.75 | 112.25 | 0% |
| | | 8 | 60 | 0.750 | 0.0648 | 123.50 | 90.13 | 0% |
| | | 8 | 100 | 0.875 | 0.0541 | 159.13 | 135.75 | 0% |

ing can introduce additional behavioral objectives beyond those explicitly specified by the original RL reward function.

**Question 2: Are frozen-backbone and LoRA-based strategies effective for preference fine-tuning of VLA policies?**

**Answer:** Yes. Both frozen-backbone and LoRA-based preference optimization can successfully modify the behavior of NaviVLA. The Qwen3.5-0.8B frozen models maintain relatively stable navigation performance across different preference objectives, with evaluation success rates generally remaining between $88\%$ and $98\%$. In contrast, the Qwen3.5-9B LoRA model exhibits stronger behavioral adaptation but also substantially larger training instability. For Long-Path DPO, the chosen metric increases from approximately $31$ to values above $60$ during training while the policy remains recoverable. However, under Long-Time DPO, the success rate decreases from $98\%$ at update 15 to $60\%$ at update 30 and eventually collapses to $0\%$ after update 45.

These results suggest that frozen tuning provides a more conservative and stable preference adaptation, whereas LoRA offers greater capacity to modify the learned policy but may introduce stronger distribution shift during online preference optimization. Since model size and the number of preference pairs also differ between these experiments, the current results demonstrate this empirical tendency rather than an isolated advantage of one tuning strategy.

**Question 3: Which factors have a significant effect on RLHF-VLA training stability?**

**Answer:** The preference objective, adaptation strategy, number of preference pairs, and quality of the initial VLA policy have substantial effects on online preference optimization. In particular, the preference target has a strong influence on policy stability. The Qwen3.5-0.8B frozen models can optimize short-time, short-path, long-time, and long-path preferences while maintaining relatively high navigation success rates. In contrast, the Qwen3.5-9B LoRA model becomes unstable under the Long-Time objective, where the preference metric continues to increase while navigation success eventually collapses to $0\%$. This indicates that successfully optimizing the preference objective does not necessarily imply preservation of the original navigation capability.

The amount of preference data per update also appears to affect optimization behavior. The frozen 0.8B experiments use $128$ preference pairs per update for most objectives and exhibit relatively smooth preference learning, whereas the 9B LoRA experiments use only $8$ pairs per update and show substantially larger fluctuations in preference accuracy, reward margin, and evaluation success rate. Overall, these results indicate that online RLHF-VLA requires a careful balance between preference strength and preservation of the original navigation policy. In particular, aggressive long-horizon preference optimization combined with highly adaptive parameter tuning can lead to preference-induced policy collapse.

# D NaviVLA Appendix

## D.1 Robot Input

### D.1.1 Multi-Camera Observation

We employ a multi-camera observation module for the NaviVLA framework. Multiple RGB-D cameras are mounted around the robot to capture visual information from different viewing directions, such as the front, left, right, and rear sides. Compared with a single-view camera, the multi-camera setting reduces blind spots and improves the robot's awareness of nearby pedestrians, obstacles, and free-space regions. The collected images are further used to construct visual observations VLA, enabling the robot to better understand dynamic human behaviors and make socially compliant navigation decisions in crowded scenarios. Each camera has about 95 degree field-of-view within the view range $[0.1m, 5m]$, as shown in Fig. 39, Fig. 40, Fig. 41, Fig. 42, and Fig. 43.

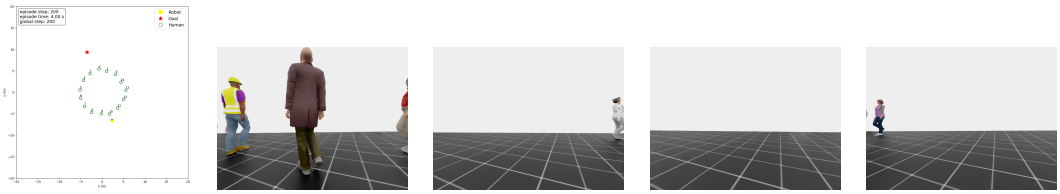
### D.1.2 BEV Map Observation

To provide the robot with a structured spatial understanding of the environment, we employ a bird's-eye-view (BEV) map observation module in the social navigation benchmark. The BEV map projects the surrounding scene onto a top-down representation, where the robot, target position, pedestrians, obstacles, and free-space regions can be encoded in a unified coordinate frame. Compared with raw visual observations, the BEV map provides clearer geometric relationships, including relative positions, motion directions, and crowd distribution. This representation helps the VLA understand the global layout and local interaction patterns, enabling safer and more socially compliant robot motion in dynamic human environments, as shown in Fig. 39, Fig. 40, Fig. 41, Fig. 42, and Fig. 43.



(a) BEV Map (b) Front Camera (c) Left Camera (d) Rear Camera (e) Right Camera

Figure 39: Visualization of the multimodal observation at step 0 for NaviVLA, including the global BEV map and four RGB camera views.



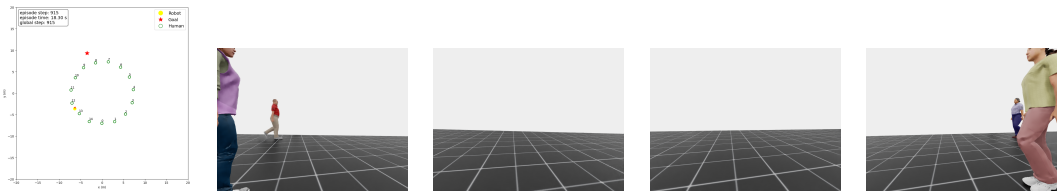
(a) BEV Map (b) Front Camera (c) Left Camera (d) Rear Camera (e) Right Camera

Figure 40: Visualization of the multimodal observation at step 200 for NaviVLA, including the global BEV map and four RGB camera views.



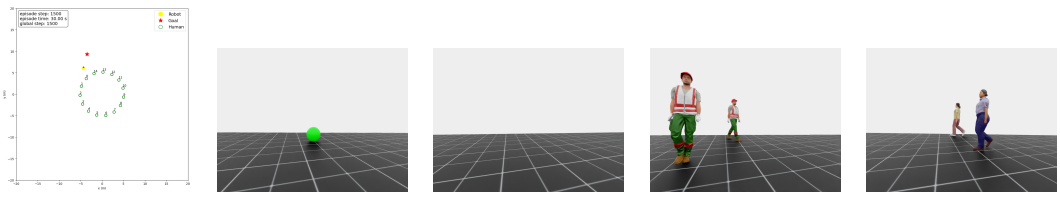
(a) BEV Map (b) Front Camera (c) Left Camera (d) Rear Camera (e) Right Camera

Figure 41: Visualization of the multimodal observation at step 500 for NaviVLA, including the global BEV map and four RGB camera views.



(a) BEV Map (b) Front Camera (c) Left Camera (d) Rear Camera (e) Right Camera

Figure 42: Visualization of the multimodal observation at step 915 for NaviVLA, including the global BEV map and four RGB camera views.



(a) BEV Map (b) Front Camera (c) Left Camera (d) Rear Camera (e) Right Camera

Figure 43: Visualization of the multimodal observation at step 1500 for NaviVLA, including the global BEV map and four RGB camera views.

### D.1.3 WORLD MODEL TEXTUAL OBSERVATION

Table 8: Logical Form Examples of Social Robot Navigation World Model

| Function (obj.) | Elements | Example Templates |
|---|---|---|
| `Vertex(robot)` | (`agent=r`, `location=p`, `velocity=v`, `target=g`, `task=T`, `distance=d`) | `[r]` is located at `[p]` with velocity `[v]` toward the destination `[g]`, executing task `[T]` with social distance `[d]`. |
| `Vertex(human)` | (`agent=h`, `location=p`, `velocity=v`, `radius=r`, `activity=a`) | `[h]` is located at `[p]` with velocity `[v]` and the collision radius `[r]`, performing personal activity `[a]`. |
| `Temporal Edge(robot)` | (`agent=r`, `target=g`, `sdistance=s`, `deltas=`$\Delta$`s`, `dynamics=yr`) | The absolute distance `[s]` between agent `[r]` and destination `[g]` has [y], compared to the last timestep with a difference `[`$\Delta$`s]`. |
| `Temporal Edge(human)` | (`agent=h`, `velocity=v`, `deltav=`$\Delta$`v`, `dynamics=yh`) | The velocity `[v]` of agent `[h]` has `[yh]` compared to the last timestep, with a difference `[`$\Delta$`v]`. |
| `Spatial Edge(human; robot)` | (`agent=h;r`, `deltard=`$\Delta$`rd`, `rdistance=rd`, `dynamics=yhr`) | The relative distance `[rd]` between `[h]` and `[r]` has `[yhr]` compared to the last timestep, with a difference `[`$\Delta$`rd]`. |

To bridge low-level geometric observations and high-level VLA reasoning, NaviVLA employs a structured social-navigation world model Wang et al. (2026). This world model converts the robot observation $o_t^r$ into language-compatible context by organizing the scene as a spatio–temporal logical graph $\mathcal{M}_t = (\mathcal{N}_t, \mathcal{R}_t)$, where $\mathcal{N}_t$ represents the robot and pedestrian nodes, and $\mathcal{R}_t$ describes their spatial and temporal relations. Compared with directly prompting the VLA with unordered numerical states, this representation provides a compact and semantically meaningful abstraction of dynamic agents, goal progress, and human–robot interactions Opedal et al. (2023); Saparov & Mitchell (2022); Xiang et al. (2024); Zhou et al. (2025).

Each node stores the observable state of an agent. The robot node contains its position, velocity, destination, physical radius, and heading direction, while each human node contains the pedestrian position, velocity, radius, and estimated activity. Notably, NaviVLA does not assume access to private human information, such as pedestrians' hidden goals or intentions. This makes the world model suitable for realistic social navigation, where the robot must infer human behavior from observable motion and surrounding context.

The relation set captures the interaction structure of the scene. Spatial relations describe the relative distance between the robot and each pedestrian, as well as whether the robot is approaching or moving away from them. Temporal relations describe how agent states evolve across consecutive timesteps. For the robot, temporal relations mainly encode goal progress, such as whether the distance to the destination is decreasing. For pedestrians, they describe motion variation and activity changes. These relations provide short-horizon cues for collision avoidance and socially compliant navigation.

As shown in Table. 5, each logical form is associated with symbolic elements and a natural-language template. These templates transform numerical observations into textual descriptions, including robot state, pedestrian state, goal-distance variation, and human–robot spatial interaction. The final NaviVLA language input is composed of the dynamic world-model description and a fixed environment summary. In this way, NaviVLA receives a physically grounded and semantically organized prompt, enabling the policy to reason jointly over robot motion, pedestrian dynamics, goal reaching, social distance, and safety constraints in dynamic crowds. The example of world model generated environmental dynamics context are shown in Fig. 44.

## Structured World Model Observation for NaviVLA

**Robot Node Information.** The robot is located at $(0.4842, -0.3186)$ with velocity $(0.8406, -0.3393)$. The destination is $(9.6552, -5.2704)$. The robot navigates among 15 pedestrians and should move toward the destination while maintaining at least $0.45$ m relative distance from pedestrians if possible.

**Robot Temporal Edge.** The robot is closer to the destination than in the previous timestep. The distance change is $-0.4374$, and the current distance to the destination is $10.4225$.

**Human–Robot Spatial Edges.**

| Human | Trend | $\Delta d$ | Relative Coordinate $(x, y)$ |
|---|---|---|---|
| 0 | Closer | $-8.7024$ | $(6.8200,\ 5.4055)$ |
| 1 | Closer | $-8.3034$ | $(3.8368,\ 7.3638)$ |
| 2 | Closer | $-8.5766$ | $(0.5641,\ 8.5580)$ |
| 3 | Farther | $+9.1916$ | $(-3.1156,\ 8.6475)$ |
| 4 | Farther | $+9.5638$ | $(-6.4294,\ 7.0802)$ |
| 5 | Farther | $+9.5721$ | $(-8.5159,\ 4.3709)$ |
| 6 | Farther | $+8.9003$ | $(-8.8956,\ 0.2885)$ |
| 7 | Farther | $+8.9159$ | $(-8.4218,\ -2.9269)$ |
| 8 | Closer | $-8.9250$ | $(-6.5688,\ -6.0421)$ |
| 9 | Closer | $-8.3502$ | $(-3.3134,\ -7.6647)$ |
| 10 | Closer | $-8.0414$ | $(-0.1766,\ -8.0395)$ |
| 11 | Closer | $-8.1353$ | $(3.3037,\ -7.4343)$ |
| 12 | Closer | $-7.8463$ | $(6.0460,\ -5.0011)$ |
| 13 | Closer | $-8.0665$ | $(7.8058,\ -2.0340)$ |
| 14 | Closer | $-8.4573$ | $(8.1749,\ 2.1672)$ |

**Human Node Information.**

| Human | Position $(x, y)$ | Velocity $(v_x, v_y)$ | Radius / Activity |
|---|---|---|---|
| 0 | $(7.3042,\ 5.0869)$ | $(-0.3891,\ -0.4544)$ | 0.3 / Walking |
| 1 | $(4.3210,\ 7.0452)$ | $(-0.1673,\ -0.5977)$ | 0.3 / Walking |
| 2 | $(1.0483,\ 8.2394)$ | $(0.0766,\ -0.6076)$ | 0.3 / Walking |
| 3 | $(-2.6314,\ 8.3289)$ | $(0.3296,\ -0.5146)$ | 0.3 / Walking |
| 4 | $(-5.9452,\ 6.7616)$ | $(0.5165,\ -0.3268)$ | 0.3 / Walking |
| 5 | $(-8.0317,\ 4.0523)$ | $(0.5807,\ -0.1517)$ | 0.3 / Walking |
| 6 | $(-8.4114,\ -0.0301)$ | $(0.5983,\ 0.2089)$ | 0.3 / Walking |
| 7 | $(-7.9376,\ -3.2455)$ | $(0.4890,\ 0.3484)$ | 0.3 / Walking |
| 8 | $(-6.0846,\ -6.3607)$ | $(0.2860,\ 0.5308)$ | 0.3 / Walking |
| 9 | $(-2.8292,\ -7.9833)$ | $(0.0062,\ 0.6183)$ | 0.3 / Walking |
| 10 | $(0.3075,\ -8.3581)$ | $(-0.1681,\ 0.5801)$ | 0.3 / Walking |
| 11 | $(3.7879,\ -7.7529)$ | $(-0.4036,\ 0.4456)$ | 0.3 / Walking |
| 12 | $(6.5301,\ -5.3197)$ | $(-0.5769,\ 0.2277)$ | 0.3 / Walking |
| 13 | $(8.2900,\ -2.3526)$ | $(-0.6020,\ 0.0283)$ | 0.3 / Walking |
| 14 | $(8.6591,\ 1.8486)$ | $(-0.5421,\ -0.3076)$ | 0.3 / Walking |

**Human Temporal Edges.** All pedestrians increase their velocity compared with the previous timestep.

| Human | Velocity Difference | Current Velocity $(v_x, v_y)$ |
|---|---|---|
| 0 | $+0.2240$ | $(-0.3891,\ -0.4544)$ |
| 1 | $+0.2331$ | $(-0.1673,\ -0.5977)$ |
| 2 | $+0.2298$ | $(0.0766,\ -0.6076)$ |
| 3 | $+0.2292$ | $(0.3296,\ -0.5146)$ |
| 4 | $+0.2292$ | $(0.5165,\ -0.3268)$ |
| 5 | $+0.2248$ | $(0.5807,\ -0.1517)$ |
| 6 | $+0.2383$ | $(0.5983,\ 0.2089)$ |
| 7 | $+0.2252$ | $(0.4890,\ 0.3484)$ |
| 8 | $+0.2259$ | $(0.2860,\ 0.5308)$ |
| 9 | $+0.2320$ | $(0.0062,\ 0.6183)$ |
| 10 | $+0.2266$ | $(-0.1681,\ 0.5801)$ |
| 11 | $+0.2250$ | $(-0.4036,\ 0.4456)$ |
| 12 | $+0.2328$ | $(-0.5769,\ 0.2277)$ |
| 13 | $+0.2259$ | $(-0.6020,\ 0.0283)$ |
| 14 | $+0.2341$ | $(-0.5421,\ -0.3076)$ |

## Structured World-Model Observation for NaviVLA

### Robot Node Information

The Robot is located at location $(0.4842, -0.3186)$ with the velocity $(0.8406, -0.3393)$ toward the destination $(9.6552, -5.2704)$.

The Robot is navigating via 15 pedestrians. Please generate robot action to go near the destination, while maintaining relative distance $0.45$ m at least with other pedestrians if it is possible.

### Robot Temporal Edge Information

The Robot's absolute distance to the destination $(9.6552, -5.2704)$ has decreased compared to the last timestep, with a difference of $(-0.4374)$, and the current absolute distance to the destination is $(10.4225)$.

### Human–Robot Spatial Edge Information

The Robot is closer toward Human0 compared to the last timestep, with a relative distance of $(-8.7024)$, and the current relative coordinate between the robot and Human0 is $(6.82, 5.4055)$.

The Robot is closer toward Human1 compared to the last timestep, with a relative distance of $(-8.3034)$, and the current relative coordinate between the robot and Human1 is $(3.8368, 7.3638)$.

The Robot is closer toward Human2 compared to the last timestep, with a relative distance of $(-8.5766)$, and the current relative coordinate between the robot and Human2 is $(0.5641, 8.558)$.

The Robot is farther toward Human3 compared to the last timestep, with a relative distance of $(9.1916)$, and the current relative coordinate between the robot and Human3 is $(-3.1156, 8.6475)$.

The Robot is farther toward Human4 compared to the last timestep, with a relative distance of $(9.5638)$, and the current relative coordinate between the robot and Human4 is $(-6.4294, 7.0802)$.

The Robot is farther toward Human5 compared to the last timestep, with a relative distance of $(9.5721)$, and the current relative coordinate between the robot and Human5 is $(-8.5159, 4.3709)$.

The Robot is farther toward Human6 compared to the last timestep, with a relative distance of $(8.9003)$, and the current relative coordinate between the robot and Human6 is $(-8.8956, 0.2885)$.

The Robot is farther toward Human7 compared to the last timestep, with a relative distance of $(8.9159)$, and the current relative coordinate between the robot and Human7 is $(-8.4218, -2.9269)$.

The Robot is closer toward Human8 compared to the last timestep, with a relative distance of $(-8.9250)$, and the current relative coordinate between the robot and Human8 is $(-6.5688, -6.0421)$.

The Robot is closer toward Human9 compared to the last timestep, with a relative distance of $(-8.3502)$, and the current relative coordinate between the robot and Human9 is $(-3.3134, -7.6647)$.

The Robot is closer toward Human10 compared to the last timestep, with a relative distance of $(-8.0414)$, and the current relative coordinate between the robot and Human10 is $(-0.1766, -8.0395)$.

The Robot is closer toward Human11 compared to the last timestep, with a relative distance of $(-8.1353)$, and the current relative coordinate between the robot and Human11 is $(3.3037, -7.4343)$.

The Robot is closer toward Human12 compared to the last timestep, with a relative distance of $(-7.8463)$, and the current relative coordinate between the robot and Human12 is $(6.046, -5.0011)$.

The Robot is closer toward Human13 compared to the last timestep, with a relative distance of $(-8.0665)$, and the current relative coordinate between the robot and Human13 is $(7.8058, -2.034)$.

The Robot is closer toward Human14 compared to the last timestep, with a relative distance of $(-8.4573)$, and the current relative coordinate between the robot and Human14 is $(8.1749, 2.1672)$.

Figure 44: Example structured world-model observation used by NaviVLA.

**Structured World-Model Observation for NaviVLA, Continued**

**Human Node Information**

Human0 is at coordinates $(7.3042, 5.0869)$ with collision radius 0.3, moving at velocity $(-0.3891, -0.4544)$, and performing Walking activity.

Human1 is at coordinates $(4.321, 7.0452)$ with collision radius 0.3, moving at velocity $(-0.1673, -0.5977)$, and performing Walking activity.

Human2 is at coordinates $(1.0483, 8.2394)$ with collision radius 0.3, moving at velocity $(0.0766, -0.6076)$, and performing Walking activity.

Human3 is at coordinates $(-2.6314, 8.3289)$ with collision radius 0.3, moving at velocity $(0.3296, -0.5146)$, and performing Walking activity.

Human4 is at coordinates $(-5.9452, 6.7616)$ with collision radius 0.3, moving at velocity $(0.5165, -0.3268)$, and performing Walking activity.

Human5 is at coordinates $(-8.0317, 4.0523)$ with collision radius 0.3, moving at velocity $(0.5807, -0.1517)$, and performing Walking activity.

Human6 is at coordinates $(-8.4114, -0.0301)$ with collision radius 0.3, moving at velocity $(0.5983, 0.2089)$, and performing Walking activity.

Human7 is at coordinates $(-7.9376, -3.2455)$ with collision radius 0.3, moving at velocity $(0.489, 0.3484)$, and performing Walking activity.

Human8 is at coordinates $(-6.0846, -6.3607)$ with collision radius 0.3, moving at velocity $(0.286, 0.5308)$, and performing Walking activity.

Human9 is at coordinates $(-2.8292, -7.9833)$ with collision radius 0.3, moving at velocity $(0.0062, 0.6183)$, and performing Walking activity.

Human10 is at coordinates $(0.3075, -8.3581)$ with collision radius 0.3, moving at velocity $(-0.1681, 0.5801)$, and performing Walking activity.

Human11 is at coordinates $(3.7879, -7.7529)$ with collision radius 0.3, moving at velocity $(-0.4036, 0.4456)$, and performing Walking activity.

Human12 is at coordinates $(6.5301, -5.3197)$ with collision radius 0.3, moving at velocity $(-0.5769, 0.2277)$, and performing Walking activity.

Human13 is at coordinates $(8.29, -2.3526)$ with collision radius 0.3, moving at velocity $(-0.602, 0.0283)$, and performing Walking activity.

Human14 is at coordinates $(8.6591, 1.8486)$ with collision radius 0.3, moving at velocity $(-0.5421, -0.3076)$, and performing Walking activity.

**Human Temporal Edge Information**

Human0's velocity has increased compared to the last timestep with the velocity difference $(+0.224)$, and current velocity is $(-0.3891, -0.4544)$.

Human1's velocity has increased compared to the last timestep with the velocity difference $(+0.2331)$, and current velocity is $(-0.1673, -0.5977)$.

Human2's velocity has increased compared to the last timestep with the velocity difference $(+0.2298)$, and current velocity is $(0.0766, -0.6076)$.

Human3's velocity has increased compared to the last timestep with the velocity difference $(+0.2292)$, and current velocity is $(0.3296, -0.5146)$.

Human4's velocity has increased compared to the last timestep with the velocity difference $(+0.2292)$, and current velocity is $(0.5165, -0.3268)$.

Human5's velocity has increased compared to the last timestep with the velocity difference $(+0.2248)$, and current velocity is $(0.5807, -0.1517)$.

Human6's velocity has increased compared to the last timestep with the velocity difference $(+0.2383)$, and current velocity is $(0.5983, 0.2089)$.

Human7's velocity has increased compared to the last timestep with the velocity difference $(+0.2252)$, and current velocity is $(0.489, 0.3484)$.

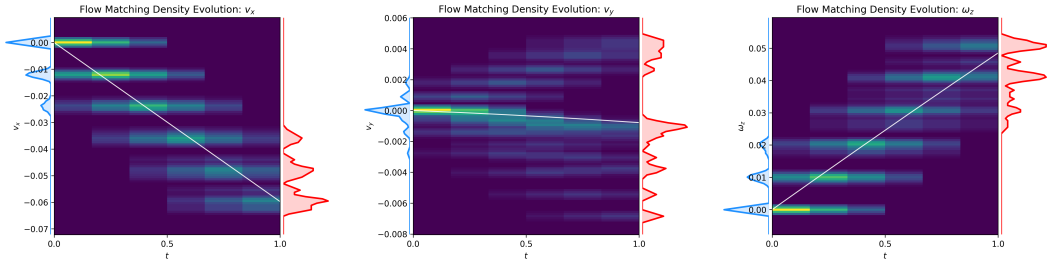
Human8's velocity has increased compared to the last timestep with the velocity difference $(+0.2259)$, and current velocity is $(0.286, 0.5308)$.

Human9's velocity has increased compared to the last timestep with the velocity difference $(+0.232)$, and current velocity is $(0.0062, 0.6183)$.

Human10's velocity has increased compared to the last timestep with the velocity difference $(+0.2266)$, and current velocity is $(-0.1681, 0.5801)$.

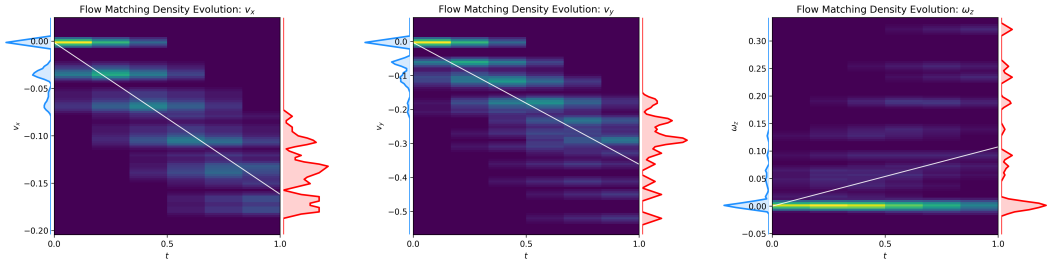
...

Figure 44: Example structured world-model observation used by NaviVLA, continued.



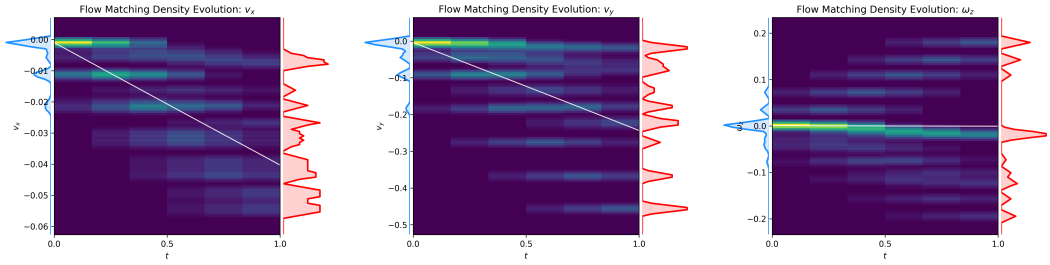

(a) Action dimension $v_x$ (b) Action dimension $v_y$ (c) Action dimension $\omega_z$

Figure 45: Visualization of the flow matching action expert. The three density plots show the learned action distributions for the commanded planar velocities $v_x$, $v_y$, and yaw angular velocity $\omega_z$.



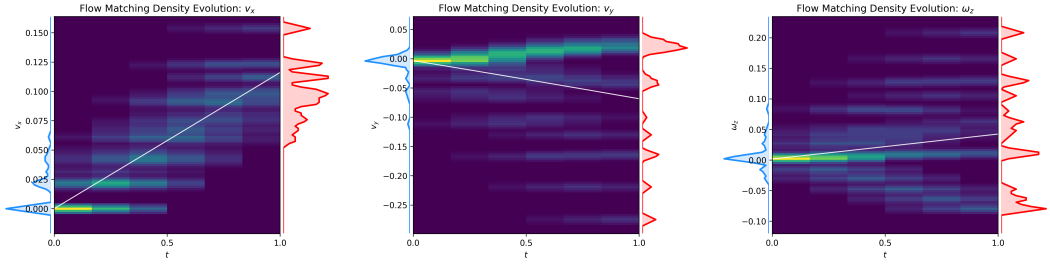

(a) Action dimension $v_x$ (b) Action dimension $v_y$ (c) Action dimension $\omega_z$

Figure 46: Flow matching action expert visualization of step-2 for the three action dimensions.




(a) Action dimension $v_x$ (b) Action dimension $v_y$ (c) Action dimension $\omega_z$

Figure 47: Flow matching action expert visualization of step-3 for the three action dimensions.




(a) Action dimension $v_x$ (b) Action dimension $v_y$ (c) Action dimension $\omega_z$

Figure 48: Flow matching action expert visualization of step-4 for the three action dimensions.

## D.2 NaviVLA RL Fine-Tuning Framework

### D.2.1 Flow Matching VLA

To decode the navigation-aware VLA feature into continuous robot commands, NaviVLA employs a PPO-compatible flow-matching action expert. Different from standard MLP action heads that directly regress the action mean, the proposed expert generates the action through an iterative vector-field refinement process in the robot velocity action space. Given the pooled VLA feature $\hat{H}_t$, the action expert first projects it into a latent flow condition $c_t$ using a lightweight projection network. Starting from a deterministic initial action state $x_{\text{init}} \in \mathbb{R}^3$, the expert repeatedly updates the intermediate action state according to a learned conditional vector field. The vector field takes the current action state, the flow condition, and a sinusoidal flow-time embedding as input, and predicts the update direction in the velocity action space.

In our implementation, the action expert is designed for one-step high-level robot control, where the action dimension is three, corresponding to $a_t^r = [v_x, v_y, \omega_z]$. To remain compatible with PPO, no random noise is sampled inside the flow expert. The initial action is deterministic, either a learnable action vector or a zero vector. After $K$ Euler-style reverse integration steps from flow time 1 to 0, the final refined action state is used as the deterministic mean of the high-level Gaussian policy. The stochasticity required for PPO exploration and policy-ratio computation is introduced externally by the Gaussian policy through the learnable log-standard-deviation parameter, as shown in Fig. 45, Fig. 46, Fig. 47, and Fig. 48.

### D.2.2 PPO for VLA

Different from standard PPO, which usually optimizes a policy over compact vector observations, NaviVLA requires policy learning from high-dimensional multimodal inputs, including multi-view camera images, BEV maps, and world-model language descriptions. Therefore, we develop a VLA-PPO training pipeline that preserves the original PPO clipped surrogate objective while redesigning the policy architecture, rollout storage, and update procedure to support VLA-based social navigation.

In standard skrl PPO, the actor typically maps a flat observation vector directly to the mean of a Gaussian action distribution. In contrast, VLA-PPO replaces the conventional MLP actor with a multimodal VLA actor. The policy first encodes multi-view visual observations, BEV context, and world-model text through a VLA encoder, compresses the resulting token sequence into a compact latent feature, and then decodes it into a high-level velocity command through a flow-matching action expert. The flow expert outputs the deterministic mean of the Gaussian policy, while the learnable standard deviation is retained for stochastic exploration and PPO probability-ratio computation. This is consistent with the modular Gaussian policy implementation, where the policy body outputs the action mean and the log-standard deviation is learned separately.

To make PPO compatible with VLA training, we further introduce a VLA-aware rollout buffer. Unlike the standard PPO memory, which stores only states, actions, rewards, dones, values, and log probabilities, the NaviVLA buffer additionally stores the aligned multimodal context at each rollout step, including front, left, rear, and right camera images, BEV images, and world-model text. This ensures that, during PPO updates, the policy can recompute action log probabilities using the same multimodal observations that were used during rollout collection.

A key modification is the multimodal re-forwarding update. During rollout collection, the VLA encoder can be executed under no grad for efficient action sampling. However, during PPO optimization, VLA-PPO does not update the policy from detached cached features. Instead, it re-forwards the stored multimodal observations through the VLA encoder, attention-pooling module, and action expert, allowing gradients to update the trainable VLA adapter, attention pooling, LoRA parameters, and flow-matching action expert. The implementation explicitly separates the rollout phase from the PPO update phase and requires navivla inputs during update to avoid using stale cached features.

To reduce the computational cost of repeatedly querying a large VLA model, VLA-PPO also introduces a high-level decision interval. The VLA policy is queried only at sparse high-level decision steps, and the generated velocity command is held for several low-level simulation steps. For example, if the decision interval is set to 25 and the environment step is 0.02 s, NaviVLA produces a new

high-level command every 0.5 s, while intermediate simulation steps reuse the previous command. This design significantly reduces VLA inference cost while preserving closed-loop PPO training.

In addition, VLA-PPO decouples actor and critic computation. The actor uses the full multimodal VLA pipeline, while the critic remains a lightweight MLP over compact state observations. This design avoids running the expensive VLA encoder for value estimation and improves training efficiency.

#### D.2.3 LoRA for VLA

To efficiently adapt the large-scale VLA backbone to social navigation, NaviVLA employs Low-Rank Adaptation (LoRA) during PPO training. Since the Qwen/VLA backbone contains a large number of parameters, directly updating the entire model with PPO would introduce high memory cost, slow training speed, and potential optimization instability. Therefore, we freeze the pretrained Qwen/VLA backbone and insert trainable low-rank adapters into its attention projection layers.

For a pretrained linear projection matrix $W$, LoRA does not directly update the full matrix. Instead, it learns a low-rank residual update:

$$W' = W + \Delta W, \qquad \Delta W = \frac{\alpha}{r} BA, \tag{57}$$

where $r$ denotes the LoRA rank, $\alpha$ is the scaling factor, and $A$ and $B$ are trainable low-rank matrices. In NaviVLA, LoRA adapters are applied to the attention projection modules of the VLA backbone, including `q_proj`, `k_proj`, `v_proj`, and `o_proj`. This allows the model to adapt its multimodal representation to the social navigation task while keeping the original backbone parameters fixed.

During NaviVLA-LoRA PPO training, the full Qwen/VLA backbone is not updated. In our implementation, parameters associated with LoRA adapters are trainable, while the remaining backbone parameters are frozen. The trainable components include the LoRA adapters, token projection layer, attention-pooling module, feature projector, flow-matching action expert, Gaussian log-standard-deviation parameter, and the critic network. The frozen components include the original Qwen/VLA backbone parameters, non-LoRA attention weights, feed-forward layers, and other pretrained vision-language model parameters.

This parameter-efficient design enables PPO to fine-tune NaviVLA through task-specific lightweight modules instead of the full VLA backbone. As a result, NaviVLA can adapt its visual-language representation to closed-loop social navigation while reducing GPU memory usage, improving training efficiency, and stabilizing reinforcement learning optimization.

#### D.2.4 Training Infrastructure

The local workstation with 4 $\times$ Nvidia RTX 4090 is used to run the IsaacSim environment and collect RL rollout databuffers, which are transferred to the training server with 8 $\times$ Nvidia H100 through the data transfer pipeline. The NaviVLA framework is then optimized on the training server using the collected rollout data. After each training epoch, the updated VLA checkpoint is synchronized back to the local workstation and loaded into the simulator to collect new data, forming an iterative data collection and policy optimization loop.

Table 9: Human Diffusion Appendix

| Method | Human Diffusion on ETH | | | | Human Diffusion on HOTEL | | | | Human Diffusion on UNIV | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR | SC | NL | NT | SR | SC | NL | NT | SR | SC | NL | NT |
| A* | 0.369 | 56.50 | 12.57 | 12.71 | 0.125 | 44.46 | 9.25 | 9.48 | 0.049 | 39.79 | 7.56 | 7.81 |
| RRT* | 0.348 | 55.13 | 12.47 | 12.69 | 0.120 | 44.15 | 9.70 | 10.02 | 0.040 | 39.57 | 7.76 | 8.13 |
| Social Force | 0.360 | 55.66 | 11.98 | 12.20 | 0.130 | 44.59 | 9.14 | 9.50 | 0.050 | 39.76 | 7.93 | 8.33 |
| ORCA | 0.440 | 59.57 | 13.44 | 14.21 | 0.220 | 48.27 | 12.52 | 13.96 | 0.120 | 43.55 | 10.43 | 11.81 |
| MLP (Rumelhart et al., 1986) | 0.023 | 3.90 | 15.19 | 14.68 | 0.066 | 7.14 | 18.07 | 17.38 | 0.058 | 6.51 | 17.51 | 16.86 |
| Transformer (Vaswani et al., 2017) | 0.085 | 8.33 | 12.65 | 11.31 | 0.175 | 14.81 | 14.99 | 13.47 | 0.158 | 13.55 | 14.54 | 13.05 |
| CADRL | 0.750 | 52.43 | 33.48 | 23.04 | 0.640 | 37.77 | 30.73 | 20.34 | 0.470 | 18.59 | 27.59 | 17.16 |
| LSTM-RL | 0.270 | 11.05 | 35.90 | 24.81 | 0.310 | 14.19 | 37.63 | 25.94 | 0.200 | 3.93 | 27.29 | 17.59 |
| SARL | 0.080 | 3.93 | 61.99 | 36.16 | 0.260 | 15.43 | 65.55 | 37.89 | 0.160 | 8.35 | 64.36 | 38.15 |
| DSRNN | 0.470 | 24.00 | 31.30 | 22.09 | 0.420 | 19.87 | 31.56 | 22.11 | 0.440 | 18.61 | 30.61 | 20.86 |
| NaviSTAR | 0.642 | 52.02 | 32.89 | 28.90 | 0.804 | 70.19 | 33.91 | 26.13 | 0.748 | 62.74 | 33.57 | 26.82 |
| NaviDIFF (Wang et al., 2025b) | 0.607 | 48.27 | 29.72 | 23.53 | 0.761 | 65.13 | 30.64 | 21.27 | 0.708 | 58.22 | 30.33 | 21.84 |
| NaviVLA wo DPO | 0.910 | 85.58 | 22.72 | 24.96 | 0.950 | 90.50 | 24.55 | 26.98 | 0.980 | 94.68 | 24.99 | 27.35 |
| **NaviVLA** | 0.905 | 91.25 | 23.25 | 26.01 | 0.935 | 94.75 | 24.96 | 27.40 | 0.980 | 101.29 | 26.05 | 28.53 |

Table 10: Human Diffusion Appendix2

| Method | Human Diffusion on ZARA1 | | | | Human Diffusion on ZARA2 | | | | Human Diffusion on SDD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR | SC | NL | NT | SR | SC | NL | NT | SR | SC | NL | NT |
| A* | 0.304 | 53.63 | 12.57 | 12.74 | 0.222 | 49.37 | 11.42 | 11.61 | 0.645 | 72.03 | 15.94 | 16.04 |
| RRT* | 0.300 | 52.67 | 13.59 | 13.92 | 0.140 | 45.03 | 10.66 | 10.97 | 0.660 | 72.83 | 15.97 | 16.13 |
| Social Force | 0.330 | 54.77 | 13.11 | 13.34 | 0.200 | 48.15 | 11.25 | 11.51 | 0.630 | 71.28 | 15.48 | 15.57 |
| ORCA | 0.500 | 63.64 | 15.97 | 16.80 | 0.350 | 55.13 | 14.74 | 15.76 | 0.810 | 81.42 | 18.12 | 18.54 |
| MLP (Rumelhart et al., 1986) | 0.048 | 5.79 | 16.87 | 16.26 | 0.047 | 5.70 | 16.79 | 16.18 | 0.022 | 3.81 | 15.11 | 14.60 |
| Transformer (Vaswani et al., 2017) | 0.137 | 12.11 | 14.02 | 12.57 | 0.135 | 11.93 | 13.95 | 12.51 | 0.082 | 8.15 | 12.58 | 11.25 |
| CADRL | 0.620 | 36.40 | 32.11 | 21.65 | 0.640 | 39.79 | 32.27 | 21.70 | 0.570 | 34.47 | 36.38 | 23.75 |
| LSTM-RL | 0.360 | 19.25 | 43.08 | 30.17 | 0.210 | 9.54 | 41.30 | 29.25 | 0.330 | 19.49 | 48.01 | 33.22 |
| SARL | 0.150 | 8.21 | 66.51 | 38.70 | 0.210 | 12.50 | 65.81 | 39.58 | 0.150 | 8.18 | 66.64 | 38.70 |
| DSRNN | 0.470 | 24.52 | 32.60 | 23.07 | 0.470 | 23.19 | 32.39 | 22.50 | 0.590 | 41.27 | 36.56 | 27.71 |
| NaviSTAR | 0.694 | 57.67 | 34.03 | 27.86 | 0.690 | 57.83 | 33.75 | 28.82 | 0.640 | 51.82 | 34.07 | 28.80 |
| NaviDIFF (Wang et al., 2025b) | 0.657 | 53.51 | 30.75 | 22.68 | 0.653 | 53.66 | 30.50 | 23.46 | 0.606 | 48.08 | 30.79 | 23.45 |
| NaviVLA wo DPO | 0.970 | 94.37 | 24.02 | 26.02 | 0.980 | 96.60 | 24.74 | 27.18 | 0.820 | 73.68 | 24.93 | 28.32 |
| **NaviVLA** | 0.958 | 102.53 | 24.91 | 27.28 | 0.968 | 100.76 | 25.36 | 27.86 | 0.815 | 78.26 | 25.81 | 28.87 |

# E EXPERIMENTS

Table 11: Human Diffusion Appendix

| Method | Human Diffusion on GCS | | | | Human Diffusion on EDIN | | | | Human Diffusion AVG | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR | SC | NL | NT | SR | SC | NL | NT | SR | SC | NL | NT |
| A* | 0.451 | 60.30 | 13.70 | 13.82 | 0.357 | 57.20 | 12.51 | 12.61 | 0.315 | 54.16 | 11.94 | 12.10 |
| RRT* | 0.510 | 63.28 | 15.04 | 15.30 | 0.354 | 56.98 | 12.62 | 12.81 | 0.309 | 53.70 | 12.23 | 12.50 |
| Social Force | 0.430 | 58.76 | 13.77 | 14.53 | 0.380 | 58.44 | 12.88 | 13.03 | 0.314 | 53.93 | 11.94 | 12.25 |
| ORCA | 0.710 | 74.06 | 18.85 | 20.76 | 0.360 | 57.34 | 12.92 | 13.32 | 0.439 | 60.37 | 14.62 | 15.64 |
| MLP (Rumelhart et al., 1986) | 0.028 | 4.35 | 15.55 | 14.99 | 0.026 | 4.09 | 15.39 | 14.89 | 0.04 | 5.16 | 16.31 | 15.73 |
| Transformer (Vaswani et al., 2017) | 0.097 | 9.15 | 12.95 | 11.58 | 0.091 | 8.77 | 12.81 | 11.46 | 0.12 | 10.85 | 13.56 | 12.15 |
| CADRL | 0.350 | 18.76 | 42.72 | 30.91 | 0.340 | 19.56 | 45.20 | 31.96 | 0.547 | 32.22 | 35.06 | 23.81 |
| LSTM-RL | 0.370 | 17.93 | 37.26 | 26.14 | 0.330 | 17.44 | 42.18 | 29.61 | 0.297 | 14.10 | 39.08 | 27.09 |
| SARL | 0.020 | 0.90 | 68.84 | 35.45 | 0.030 | 1.53 | 81.46 | 41.89 | 0.133 | 7.38 | 67.64 | 38.32 |
| DSRNN | 0.550 | 31.11 | 33.66 | 24.06 | 0.460 | 27.74 | 36.25 | 27.47 | 0.484 | 26.29 | 33.12 | 23.73 |
| NaviSTAR | 0.660 | 53.50 | 34.04 | 28.25 | 0.660 | 53.50 | 34.04 | 28.25 | 0.692 | 57.41 | 33.79 | 27.98 |
| NaviDIFF (Wang et al., 2025b) | 0.624 | 49.64 | 30.76 | 23.00 | 0.624 | 49.64 | 30.76 | 23.00 | 0.655 | 53.27 | 30.53 | 22.78 |
| NaviVLA wo DPO | 0.880 | 80.81 | 22.64 | 25.15 | 0.910 | 81.48 | 23.09 | 24.81 | 0.925 | 87.21 | 23.96 | 26.35 |
| **NaviVLA** | 0.869 | 91.08 | 23.20 | 25.68 | 0.894 | 90.47 | 24.17 | 25.84 | 0.906 | **93.80** | 24.71 | 27.18 |

Table 12: Crowd Simulator Appendix

| Method | Circle Crowd & Social Force | | | | Circle Crowd & ORCA | | | | Random Crowd & Social Force | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR | SC | NL | NT | SR | SC | NL | NT | SR | SC | NL | NT |
| A* | 0.043 | 27.94 | 6.28 | 6.73 | 0.017 | 33.34 | 9.93 | 10.60 | 0.091 | 37.70 | 8.68 | 9.05 |
| RRT* | 0.162 | 46.72 | 5.79 | 6.16 | 0.116 | 43.19 | 5.51 | 6.00 | 0.295 | 53.18 | 5.77 | 6.11 |
| Social Force | 0.098 | 43.08 | 5.28 | 5.64 | 0.070 | 41.03 | 4.79 | 5.19 | 0.275 | 51.79 | 5.19 | 5.57 |
| ORCA | 0.350 | 53.57 | 25.05 | 32.77 | 0.530 | 62.34 | 25.97 | 33.15 | 0.350 | 53.65 | 21.93 | 28.31 |
| MLP (Rumelhart et al., 1986) | 0.040 | 4.97 | 10.15 | 9.23 | 0.044 | 5.21 | 10.34 | 9.41 | 0.057 | 6.09 | 11.06 | 10.07 |
| Transformer (Vaswani et al., 2017) | 0.060 | 7.73 | 11.37 | 11.08 | 0.065 | 8.06 | 11.60 | 11.29 | 0.085 | 9.27 | 12.42 | 12.06 |
| CADRL | 0.400 | 11.59 | 28.64 | 17.69 | 0.430 | 12.77 | 29.15 | 18.73 | 0.420 | 14.83 | 28.97 | 18.75 |
| LSTM-RL | 0.410 | 16.99 | 34.48 | 23.18 | 0.470 | 21.22 | 34.49 | 23.27 | 0.560 | 30.36 | 34.36 | 21.92 |
| SARL | 0.460 | 24.63 | 42.29 | 28.22 | 0.550 | 31.70 | 43.61 | 29.09 | 0.370 | 18.62 | 45.11 | 28.11 |
| DSRNN | 0.330 | 7.95 | 24.90 | 14.44 | 0.510 | 23.52 | 31.92 | 21.50 | 0.360 | 11.06 | 25.13 | 15.63 |
| NaviSTAR | 0.740 | 53.88 | 36.25 | 28.94 | 0.759 | 53.80 | 36.01 | 28.37 | 0.820 | 68.31 | 33.00 | 27.11 |
| NaviDIFF (Wang et al., 2025b) | 0.762 | 56.15 | 42.42 | 33.30 | 0.782 | 56.07 | 42.14 | 32.64 | 0.844 | 71.19 | 38.62 | 31.19 |
| NaviVLA wo DPO | 0.820 | 62.52 | 28.38 | 31.19 | 0.770 | 51.42 | 26.73 | 29.48 | 0.790 | 63.32 | 25.90 | 29.51 |
| **NaviVLA** | 0.794 | 73.67 | 29.49 | 31.79 | 0.741 | 57.06 | 27.15 | 30.62 | 0.769 | 71.95 | 27.03 | 30.23 |

Table 13: Crowd Simulator Appendix

| Method | Random Crowd & ORCA | | | | Corridor Crowd & Social Force | | | | Corridor Crowd & ORCA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR | SC | NL | NT | SR | SC | NL | NT | SR | SC | NL | NT |
| A* | 0.065 | 37.24 | 8.47 | 8.88 | 0.066 | 35.24 | 8.09 | 8.69 | 0.517 | 63.64 | 16.86 | 17.15 |
| RRT* | 0.262 | 50.53 | 5.20 | 5.54 | 0.329 | 56.48 | 6.61 | 6.95 | 0.469 | 65.20 | 7.07 | 7.38 |
| Social Force | 0.268 | 51.02 | 4.96 | 5.34 | 0.336 | 57.06 | 6.35 | 6.66 | 0.425 | 62.75 | 6.47 | 6.78 |
| ORCA | 0.360 | 54.53 | 21.32 | 27.16 | 0.210 | 48.04 | 23.45 | 33.12 | 0.330 | 54.92 | 24.64 | 30.47 |
| MLP (Rumelhart et al., 1986) | 0.045 | 5.29 | 10.41 | 9.47 | 0.016 | 3.37 | 8.85 | 8.03 | 0.020 | 3.61 | 9.04 | 8.21 |
| Transformer (Vaswani et al., 2017) | 0.067 | 8.17 | 11.67 | 11.36 | 0.024 | 5.53 | 9.87 | 9.68 | 0.029 | 5.86 | 10.09 | 9.89 |
| CADRL | 0.540 | 23.99 | 28.41 | 18.17 | 0.040 | 0.07 | 11.31 | 3.24 | 0.070 | 0.20 | 12.59 | 4.13 |
| LSTM-RL | 0.520 | 26.39 | 32.73 | 20.76 | 0.080 | 0.36 | 11.77 | 3.41 | 0.160 | 1.56 | 13.55 | 5.45 |
| SARL | 0.350 | 17.66 | 44.45 | 27.32 | 0.200 | 3.22 | 19.36 | 10.35 | 0.280 | 5.57 | 18.64 | 11.07 |
| DSRNN | 0.290 | 8.16 | 24.08 | 15.10 | 0.040 | 0.08 | 12.08 | 3.82 | 0.140 | 1.73 | 16.82 | 9.16 |
| NaviSTAR | 0.760 | 56.92 | 33.76 | 26.61 | 0.420 | 16.00 | 20.92 | 14.05 | 0.442 | 13.61 | 22.65 | 15.55 |
| NaviDIFF (Wang et al., 2025b) | 0.783 | 59.32 | 39.51 | 30.62 | 0.432 | 16.67 | 24.48 | 16.17 | 0.455 | 14.18 | 26.51 | 17.89 |
| NaviVLA wo DPO | 0.740 | 52.47 | 24.09 | 27.54 | 0.500 | 19.69 | 15.56 | 17.32 | 0.390 | 10.80 | 12.03 | 13.87 |
| **NaviVLA** | 0.714 | 57.95 | 24.52 | 28.02 | 0.489 | 29.30 | 16.05 | 17.90 | 0.371 | 20.04 | 12.45 | 14.95 |

Table 14: Crowd Simulator Appendix

| Method | Group Crowd & Social Force | | | | Group Crowd & ORCA | | | | Crowd Simulator AVG | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR | SC | NL | NT | SR | SC | NL | NT | SR | SC | NL | NT |
| A* | 0.085 | 38.32 | 8.45 | 8.76 | 0.195 | 45.31 | 10.81 | 11.08 | 0.135 | 39.84 | 9.70 | 10.12 |
| RRT* | 0.381 | 58.30 | 6.42 | 6.74 | 0.526 | 67.56 | 7.37 | 7.66 | 0.318 | 55.14 | 6.22 | 6.57 |
| Social Force | 0.384 | 58.66 | 6.14 | 6.49 | 0.531 | 67.88 | 7.20 | 7.56 | 0.298 | 54.16 | 5.80 | 6.15 |
| ORCA | 0.210 | 44.72 | 16.90 | 22.24 | 0.450 | 59.98 | 19.94 | 24.81 | 0.349 | 53.97 | 22.40 | 29.00 |
| MLP (Rumelhart et al., 1986) | 0.052 | 5.77 | 10.80 | 9.83 | 0.047 | 5.45 | 10.54 | 9.59 | 0.04 | 4.97 | 10.15 | 9.23 |
| Transformer (Vaswani et al., 2017) | 0.078 | 8.83 | 12.12 | 11.78 | 0.071 | 8.39 | 11.82 | 11.50 | 0.06 | 7.73 | 11.37 | 11.08 |
| CADRL | 0.460 | 18.79 | 29.18 | 19.45 | 0.530 | 23.22 | 28.09 | 18.37 | 0.361 | 13.18 | 24.54 | 14.82 |
| LSTM-RL | 0.490 | 23.05 | 29.92 | 19.08 | 0.520 | 26.02 | 31.89 | 19.86 | 0.401 | 18.24 | 27.90 | 17.12 |
| SARL | 0.270 | 11.53 | 42.61 | 25.74 | 0.330 | 15.88 | 44.75 | 26.60 | 0.351 | 16.10 | 37.60 | 23.31 |
| DSRNN | 0.280 | 8.54 | 25.63 | 16.82 | 0.360 | 14.11 | 26.95 | 18.73 | 0.289 | 9.39 | 23.44 | 14.40 |
| NaviSTAR | 0.800 | 62.12 | 31.71 | 24.68 | 0.775 | 60.10 | 33.78 | 26.67 | 0.690 | 48.09 | 31.01 | 24.00 |
| NaviDIFF (Wang et al., 2025b) | 0.824 | 64.74 | 37.11 | 28.40 | 0.798 | 62.63 | 39.53 | 30.68 | 0.710 | 50.12 | 36.29 | 27.61 |
| NaviVLA wo DPO | 0.770 | 59.19 | 25.47 | 29.07 | 0.770 | 59.38 | 24.92 | 28.91 | 0.694 | 47.35 | 22.88 | 25.86 |
| **NaviVLA** | 0.757 | 72.52 | 26.02 | 30.20 | 0.763 | 72.77 | 25.43 | 29.78 | 0.674 | **56.91** | 23.52 | 26.69 |

## F ACKNOWLEDGE

In this manuscript, generative AI tools were used to assist with figure generation, mainly for Figure 2. In addition, AI tools were used to improve the writing quality and clarity of the manuscript. All AI-assisted content was carefully reviewed, verified, and edited by the authors.

More details of this paper can be found in the project website: `https://sites.google.com/view/NaviVLA`.