A Robust Conformal Uncertainty Handling Framework for Safe Crowd Navigation

Jianpeng Yao Xiaopan Zhang[†] Yu Xia[†] Zejin Wang Amit K. Roy-Chowdhury Jiachen Li[*]
University of California, Riverside

Abstract—**Mobile robots navigating in crowds trained using reinforcement learning are known to suffer performance degradation when faced with out-of-distribution scenarios. We propose that by properly accounting for the uncertainties of pedestrians, a robot can learn safe navigation policies that are robust to distribution shifts. Our method augments agent observations with prediction uncertainty estimates generated by adaptive conformal inference, and it uses these estimates to guide the agent's behavior through constrained reinforcement learning. The system helps regulate the agent's actions and enables it to adapt to distribution shifts. In the in-distribution setting, our approach achieves a 96.93% success rate, which is over 8.80% higher than the previous state-of-the-art baselines with over 3.72 times fewer collisions and 2.43 times fewer intrusions into ground-truth human future trajectories. In three out-of-distribution scenarios, our method shows much stronger robustness when facing distribution shifts in velocity variations, policy changes, and transitions from individual to group dynamics. We deploy our method on a real robot, and experiments show that the robot makes safe and robust decisions when interacting with both sparse and dense crowds.**

I. Introduction

Safely navigating among crowds is fundamental to a future where robots work closely with humans [6, 28]. Various solutions have been studied, including rule-based methods [34, 12], optimization-based planners [11, 30, 16], reinforcement learning (RL) approaches [4, 25], and hybrid systems that track a reference path from an optimizer while a learned policy makes necessary adaptations [35, 36]. For RL approaches, a single millisecond-level forward pass can generate decisions, making them faster and more scalable than optimization- or rule-based planners. Yet while they excel in in-distribution settings, their performance drops sharply in out-of-distribution (OOD) environments. This indicates a tendency towards overfitting and difficulty in generalizing to diverse crowd dynamics.

Moreover, previous studies have shown that incorporating predictions into observations, thus forming prediction-augmented observations [26, 37], can explicitly represent human intentions and aid in robot decision making. However, this practice may exacerbate overfitting issues, as human dynamics in real-world scenarios are inherently complex and difficult to fully capture in simulation environments or datasets, and trajectory predictions fitted for one certain dynamics face generalization challenges to other dynamics. Inaccurate predictions can severely mislead robot decisions, especially when robots rely heavily on them to determine actions. Due to compounded overfitting from both predictions and the learned policy, robots specialized for these environments often fail to generalize to new scenarios with different crowd dynamics. In other words, existing methods lack a systematic treatment of prediction errors to improve policy generalizability.

In this paper, we propose that by properly quantifying uncertainties in human trajectory predictions and incorporating the results into decision making algorithms, it is possible to alleviate overfitting in RL-based crowd navigation. Uncertainty serves as an indicator of the reliability of prediction hypotheses, reflecting both prediction errors and the sensitivity of prediction models to distribution shifts. Therefore, encouraging the agent to account for uncertainty enables it to generate actions that are more robust to distribution shifts and resilient to incorrect assumptions about human dynamics. Specifically, we introduce a learning-based framework that explicitly reasons about prediction uncertainty. First, we apply adaptive conformal inference (ACI) [7, 8] to quantify the uncertainty of each predicted human trajectory with a prediction set that contains the true future position with a user-defined coverage probability. Unlike other conformal approaches [1], ACI updates its calibration online, so it can swiftly adapt when the underlying crowd dynamics shift. Second, we employ constrained reinforcement learning (CRL) to introduce effective controllability into the decision making system, using uncertainty estimates to guide both the learning process and the agent's behavior. Our system achieves the state-of-the-art (SOTA) performance in safety metrics and with much smaller performance drops in OOD settings. In the in-distribution setting, our system achieves an over 8.80% higher success rate than previous SOTA RL baselines, with more than 3.72 times fewer collisions and over 2.43 times fewer intrusions into pedestrian trajectories in in-distribution settings. In three different OOD test scenarios that introduce velocity shifts, policy shifts, and pedestrian grouping behavior, our method maintains a high success rate and low collision rate, while competing approaches degrade significantly. Finally, we deploy the learned policy on a real Mecanum-wheel robot. With only minor clipping and smoothing, the policy transfers directly from simulation and achieves safe navigation in both sparse and dense crowds.

II. Related Work

Crowd Navigation. Mobile robots are expected to interact with humans and complete various tasks, such as providing assistance [6]. Crowd navigation forms the foundation for

[*] Correspondence to: Jiachen Li <`jiachen.li@ucr.edu`>
[†] These authors contributed equally to this paper.

performing most high-level tasks. Robots are required to navigate in crowds, where the challenge of modeling dynamic human behavior makes navigation challenging. It is crucial to capture the subtleties of human behavior, such as human intentions and interactions between agents [20, 9, 39, 19, 21], and properly use them for effective robotic decisions. Deep RL offers a potentially viable solution to the challenging navigation task [23, 22, 27]. Previous work on RL-based methods for social robots includes the capture of agent-agent interactions [3, 22] and the intentions of human agents [26], incorporating these as predictions into RL policy networks. Our work focuses on alleviating performance degradation in OOD scenarios and proposes a system that works towards this challenge by effectively considering and handling uncertainties in human behavior predictions.

Planning Under Uncertainty. Trajectory planning under uncertainty has attracted increasing attention. In optimization-based and search-based methods, researchers have integrated uncertainty quantification from perception and prediction into various controllers [38, 5, 15, 24]. These methods easily support adding constraints or safety shields, allowing explicit management of uncertainty. By contrast, guiding decision making in RL agents with uncertainty measures is less straightforward, since policy networks often behave as black boxes. Prior work has augmented agent observations with uncertainty estimates [13, 10] or post-processed policy outputs based on uncertainty to ensure safe behavior [33]. However, most approaches avoid integrating uncertainty guidance directly into the learning process and still suffer from OOD performance degradation similar to the methods without uncertainty measures. In our work, we propose that uncertainty-aware planning can effectively mitigate OOD performance degradation for RL-based crowd navigation and introduce a CRL framework that enhances controllability and leverages uncertainty estimates to effectively guide decision making, thereby improving robustness to prediction errors and OOD scenarios.

III. Method

A. Problem Formulation

In our setting, we have H humans in the environment, each indexed by h, within an episode of horizon T. At each time step t, the positions of humans are represented as $\mathbf{p}_h(t)$. We predict K future steps (a larger K means more extended future predictions) for each human's future trajectories. The prediction point for the k-th prediction of the h-th human is denoted as $\mathbf{p}_{h,k}(t)$, where $1 \leq h \leq H$ and $1 \leq k \leq K$. We formulate the task as a constrained Markov decision process, where the CRL agent is provided with observations of the state S_t at each timestep, which consists of two main parts. The first part includes physical information: the current positions of humans and the robot, and other quantities about the robot's dynamics. The second part comprises post-processed features generated by models, such as the predicted human trajectories. We denote physical state components of ego information as $\mathbf{e}$, physical components of human information as $\mathbf{h}$, and components generated by models as $\mathbf{m}$. The complete state S_t is written as $S_t = [\mathbf{e}, \mathbf{h}, \mathbf{m}]$. The agent then generates action $A_t = (v_x, v_y)$ to control the moving speed of the robot according to S_t. After taking an action, the environment provides a reward R_t and a cost C_t. We aim to obtain an optimal policy $\pi(A_t \mid S_t)$ that maximizes rewards while satisfying cost constraints.

B. Method Overview

The overall pipeline of our method is illustrated in Fig. 1. First, we employ two different trajectory predictors: the constant velocity (CV) predictor [31] and the Gumbel social transformer (GST) predictor [14]. This is to demonstrate that our algorithm can adapt to both rule-based and learning-based predictors. Next, we use ACI to quantify prediction uncertainties and incorporate them into $\mathbf{m}$; subsequently, we adopt a policy network with a combined attention mechanism as proposed in [26] to process these uncertainties along with other features. Finally, we employ CRL to guide the agents' behavior using the uncertainty quantification results. Instead of directly applying constraints to the collision rate, we impose constraints on the average cumulative intrusions of the robot into other agents' uncertainty areas. This approach provides behavior-level guidance and effectively addresses the issue of sparse constraint feedback, thereby improving upon previous methods that directly constrained result-oriented metrics such as the collision rate.

C. Rule-Based and Learning-Based Trajectory Prediction

Our system adapts to different prediction models, including both learning-based and rule-based trajectory predictors, and can mitigate the adverse effects of incorrect predictions on the subsequent decision-making process. For the rule-based prediction model, we use the CV model to obtain simple and effective estimates of future human states by extrapolating the current states of human agents based on their velocities. For learning-based prediction models, we choose the GST predictor [14] as the learning-based prediction model, which is designed to address the challenges of partially detected pedestrians and redundant interaction modeling. GST is adapted for the efficient encoding of pedestrian features and demonstrates adaptability and robustness in high-density crowds.

D. ACI for Quantifying Prediction Uncertainty

After obtaining the K-step future predictions, we quantify the prediction uncertainty using dynamically-tuned adaptive conformal inference (DtACI) [8], an ACI algorithm that adapts effectively to distribution shifts and thus serves as a good fit for online uncertainty estimation in social navigation. For each prediction step of each pedestrian, we run M prediction error estimators simultaneously. At time step t, we calculate the actual prediction error $\delta_{h,k}$ between the current position and the predicted position made at time step $t-k$ for the k-th prediction step of the h-th human as $\delta_{h,k}(t) = \|\mathbf{p}_h(t) - \mathbf{p}_{h,k}(t-k)\|_2$, where $\mathbf{p}_h(t)$ is the position of the h-th human at time t, $\mathbf{p}_{h,k}(t-k)$ is the k-step predicted position of the h-th human made at time $t-k$, and $\delta_{h,k}(t)$ is calculated as the

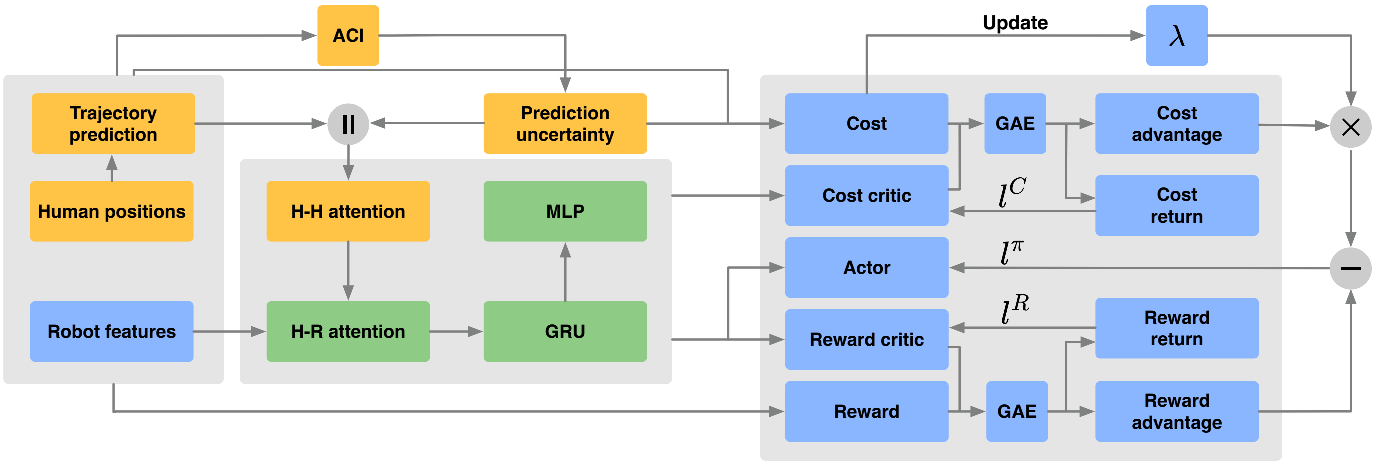

Fig. 1: The overall pipeline of our method. We mark components related to humans in yellow, components related to physical information and decision-making of the robot in blue, and fused features in green. We use ACI to quantify the prediction uncertainty of human trajectories and concatenate these metrics with predictions before inputting them into networks. The networks contain attention mechanisms for interactions between humans (H-H attention) and between humans and the ego robot (H-R attention). Prediction uncertainty combined with physical information is used for designing costs. For the CRL agent using PPO Lagrangian, the actor and reward critic share some layers while the cost critic uses a separate network. We adopt reward value loss l^R, action loss l^π, and cost value loss l^C for updating the agent.

L2 norm of the difference of $\mathbf{p}_h(t)$ and $\mathbf{p}_{h,k}(t-k)$. We then update the estimated error generated by the m-th estimator for the h-th human as

$$\hat{\delta}^{(m)}_{h,k}(t) = \hat{\delta}^{(m)}_{h,k}(t-1) \; - \; \gamma^{(m)}\left(\alpha - \mathrm{err}^{(m)}_{h,k}(t)\right), \qquad (1)$$

where $\hat{\delta}^{(m)}_{h,k}$ represents the estimated prediction error of the m-th estimator corresponding to a k-step ahead prediction for the h-th human, $\gamma^{(m)}$ is the learning rate of the m-th prediction error estimator for all humans and predictions, α is the coverage parameter, and

$$\mathrm{err}^{(m)}_{h,k}(t) := \begin{cases} 1, & \text{if } \hat{\delta}^{(m)}_{h,k}(t-1) < \delta_{h,k}(t), \\ 0, & \text{if } \hat{\delta}^{(m)}_{h,k}(t-1) \geq \delta_{h,k}(t). \end{cases} \qquad (2)$$

According to quantile regression [8], $\hat{\delta}^{(m)}_{h,k}$ converges to the $(1-\alpha)$ quantile of actual errors. Since we run M prediction error estimators with different learning rates simultaneously, for each error estimator, after taking in the actual prediction error $\delta_{h,k}$ and updating the estimated prediction error for the next step, we evaluate the errors of each estimator and update the probability distribution for choosing the next output prediction error estimator by

$$w^{(m)}_{h,k} \leftarrow (1-\sigma)\frac{w^{(m)}_{h,k}\exp\left(-\eta\, \ell(\delta_{h,k}, \delta^{(m)}_{h,k})\right)}{\sum_{j=1}^{M} w^{(j)}_{h,k}\exp\left(-\eta\, \ell(\delta_{h,k}, \delta^{(j)}_{h,k})\right)} \; + \; \frac{\sigma}{M}, \quad (3)$$

$$p^{(m)}_{h,k} \leftarrow \frac{w^{(m)}_{h,k}}{\sum_{j=1}^{M} w^{(j)}_{h,k}}, \qquad (4)$$

where we ignore the explicit time-step notation t for simplicity, and use the arrow to indicate the replacement of values. The weight $w^{(m)}_{h,k}$ corresponds to the probability $p^{(m)}_{h,k}$ for the m-th prediction error estimator, σ and η are hyperparameters of DtACI for adjusting the speed at which weights change. $\ell(\delta_{h,k}, \delta^{(m)}_{h,k})$ is the pinball loss function used to measure the estimation error:

$$\ell(\delta_{h,k}, \delta^{(m)}_{h,k}) = \begin{cases} \alpha\left(\delta_{h,k} - \delta^{(m)}_{h,k}\right), & \text{if } \delta_{h,k} \geq \delta^{(m)}_{h,k}, \\ (\alpha - 1)\left(\delta_{h,k} - \delta^{(m)}_{h,k}\right), & \text{if } \delta_{h,k} < \delta^{(m)}_{h,k}. \end{cases} \qquad (5)$$

Each time we estimate the prediction uncertainty, we treat $\hat{\delta}_{h,k}$ as a discrete random variable taking values in the set $\{\hat{\delta}^{(1)}_{h,k}, \ldots, \hat{\delta}^{(M)}_{h,k}\}$. The probability mass function of $\hat{\delta}_{h,k}$ is:

$$P\left(\hat{\delta}_{h,k} = \hat{\delta}^{(m)}_{h,k}\right) \; = \; p^{(m)}_{h,k}, \quad m = 1, \ldots, M, \quad \textstyle\sum_{m=1}^{M} p^{(m)}_{h,k} \; = \; 1. \; (6)$$

In other words, $\hat{\delta}_{h,k}$ follows a categorical distribution whose probability mass function is $\{p^{(1)}_{h,k}, \ldots, p^{(M)}_{h,k}\}$.

E. *Policy Network Structure*

Once we have obtained the trajectory prediction results and the corresponding prediction uncertainty, we concatenate the uncertainty quantification with the predicted trajectory before feeding it into the attention layers. This allows the RL agents to account for the prediction uncertainty in their decision-making process, as shown in Fig. 1. The first block is the human-human attention (H-H attention in Fig. 1), which models each human as a separate node to capture interactions among humans. Next, we fuse the robot features (including velocity, heading, positions, and goal) into the attention blocks through human-robot attention (H-R attention) to obtain fused feature embeddings capturing the interactions between humans and the ego robot. We then process these fused embeddings with the robot features and concatenate them to form the GRU input, thereby capturing temporal information. Lastly, the final fused features are passed to the actor and critic networks for further processing. For more details about the policy network structure, please refer to [26].

F. Decision Making with Effective Uncertainty Handling

To enhance controllability and enable effective guidance under uncertainty, we adopt a physically meaningful cost and adjust its distribution based on a predefined cost limit using CRL, which is not achievable with traditional RL and reward shaping, as conventional rewards provide only an aggregate signal without any mechanism to ensure that safety-related components converge within a specified range. First, we design the safety critical area of pedestrians as a combination of a circular area around the human's position and an uncertainty area around K' ($K' \leq K$) steps of predictions. Since we have H human agents in the environment, the two parts of the area are defined as

$$D_i(\mathbf{p}_{\text{ego}}) = \{\mathbf{p}_{\text{ego}} : |\mathbf{p}_{\text{ego}} - \mathbf{p}| \leq r_i\}, \ \mathbf{p} \in P_i, \ i = 1, 2, \quad (7)$$

$$P_1 = \{\mathbf{p}_h\}, \ P_2 = \{\mathbf{p}_{h,k}\}, \ 1 \leq h \leq H, \ 1 \leq k \leq K', \quad (8)$$

$$r_1 = r_{\text{ego}} + r_h + r_{\text{comfort}}, \quad r_2 = r_{\text{ego}} + r_h + \hat{\delta}_{h,k}, \quad (9)$$

where D_1 is the subarea around the current positions of humans and D_2 is the subarea around the predicted positions of humans. If the current center position of the ego robot $\mathbf{p}_{\text{ego}}$ is in either D_1 or D_2, an intrusion occurs. For the computation, we consider the distance between the center positions of agents and prediction points. r_{ego} is the radius of the ego robot, r_h is the radius of the h-th human, r_{comfort} is the radius of the subarea around the current positions of humans, and $\hat{\delta}_{h,k}$ is the prediction uncertainty generated by DtACI for the k-th prediction point of the h-th human.

At each time step t, we iterate through all cost areas of all humans and calculate the maximum intrusion, denoted as $d_{\text{intru},t}$. For an episode with a horizon of T, we have

$$\max_{\pi} \sum_{t=0}^{T} R_t(S_t, A_t) \quad \text{s.t.} \quad \sum_{t=0}^{T} d_{\text{intru},t} = \tilde{d}, \quad (10)$$

where $\tilde{d}$ is a pre-defined threshold.

We formulate the cost C_t using the intrusions as $C_t(S_t, A_t) = \mu d_{\text{intru},t}$, where μ is a constant.Our reward includes three components: R_{success}, $R_{\text{collision}}$, and $R_{\text{potential}}$. R_{success} is the reward for robot reaches the goal, $R_{\text{collision}}$ is for the robot collides with pedestrians, and $R_{\text{potential}}$ provides a dense reward that drives the ego robot to approach the goal, proportional to the distance the ego robot approaches the goal compared to the previous time step [26].

In our work, we use the PPO Lagrangian [29] for optimization. We set up two critics to compute the state value for reward and the state value for cost. The loss functions for the critics are defined as

$$l_t^R = c_1(V_{\theta_1}^R(S_t) - V_t^{\text{targ},R})^2, \quad (11)$$

$$l_t^C = c_2(V_{\theta_2}^C(S_t) - V_t^{\text{targ},C})^2, \quad (12)$$

where c_1 and c_2 are constants, $V_{\theta_1}^R(S_t)$ and $V_{\theta_2}^C(S_t)$ are network-generated value estimates for reward and cost, respectively, and $V_t^{\text{targ},R}$ and $V_t^{\text{targ},C}$ are the target values.

As for the policy network, the action loss is similar to the form in PPO [32] except that we employ the combined advantage $\hat{A}'_t = \frac{\hat{A}_t^R - \lambda \hat{A}_t^C}{1+\lambda}$. The action loss function can then be written as

$$l_t^\pi = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta_3)\hat{A}'_t, \text{clip}\left(r_t(\theta_3), 1-\epsilon, 1+\epsilon\right) \hat{A}'_t \right) \right], \quad (13)$$

where $r_t(\theta_3)$ represents the change ratio between the updated and old policies, ϵ is the clip ratio, and $\hat{A}'_t$ represents the estimated value advantage function at time step t. Lastly, we update λ using gradient descent. The loss function [17] for updating λ is defined as $l_t^\lambda = -\lambda(\bar{C} - \tilde{d}_C)$, where $\bar{C}$ is the mean episodic cost and $\tilde{d}_C$ is the *cost limit* proportional to $\tilde{d}$. Intuitively, when $\bar{C}$ is greater than $\tilde{d}_C$ (i.e., intrusions are frequent), λ will increase according to gradient descent, leading the RL agents to consider cost advantages more when updating the policy and thus, actions with larger costs will be less preferred, and vice versa.

IV. Experiments

A. Simulation Settings and Evaluation Metrics

We adopt CrowdNav [3], one of the most widely adopted simulation environments for crowd navigation. The environment consists of 20 humans and one robot in a $12\,\text{m} \times 12\,\text{m}$ area, with randomized initial positions and goal locations. The robot has a radius of $0.2\,\text{m}$, while human radii are randomly sampled between $0.3\,\text{m}$ and $0.5\,\text{m}$. The robot's maximum speed is set to $1.0\,\text{m/s}$. We adopt the standard evaluation metrics [26], including success rate (SR), collision rate (CR), timeout rate (TR), navigation time (NT), path length (PL), intrusion time ratio (ITR), and social distance (SD). The detailed definitions can be found in Appendix C. For all of these metrics, SR reflects the effectiveness and CR evaluates the safety of algorithms. TR, NT, and PL measure the efficiency of the policy-generated paths, and ITR and SD measure the politeness of the paths. We believe SR and CR are the most important metrics. Policies with lower SR or higher CR, even if they might perform better on other metrics, are not as good as those with higher SR and lower CR. We include our implementation details in Appendix D.

B. Baselines and Ablation Models

Our baselines include optimal reciprocal collision avoidance (ORCA) [34], social force (SF) [12], model-predictive control (MPC), SafeCrowdNav [37], and CrowdNav++ [26]. ORCA and SF are classic rule-based algorithms in obstacle avoidance, MPC is an optimization-based algorithm, and SafeCrowdNav and CrowdNav++ represent the previous SOTA RL algorithms in crowd navigation. For MPC, we implement a cost function similar to that in [24, 15], which uses uncertainty as a reference in the cost formulation to improve safety. We test our method with the CV and the GST predictors, represented as Ours (w/ CV) and Ours (w/ GST), respectively. We also include RL (w/o ACI) and RL (w/ ACI) as two ablation models. The two variants have the same network structures as Ours (w/ GST), but RL (w/o ACI) excludes uncertainty estimates

TABLE I: In-Distribution Test Results

Methods	**SR↑**	**CR↓**	**TR↓**	**NT↓**	**PL↓**	**ITR↓**	**SD↑**
SF	15.60%	21.44%	62.96%	30.23	34.64	3.78%	0.42
ORCA	67.84%	27.52%	4.64%	22.80	19.74	**1.10**%	**0.50**
MPC	73.76%	25.52%	0.72%	19.09	20.88	11.91%	0.43
SafeCrowdNav	89.09±3.18%	10.91±3.18%	**0.00±0.00**%	13.06±0.34	**12.06±0.45**	14.09±0.91%	0.39±0.01
CrowdNav++	86.11±0.61%	13.81±0.57%	0.00±0.14%	14.96±1.07	20.68±0.88	6.61±0.98%	0.43±0.01
RL (w/o ACI)	92.67±1.51%	7.33±1.51%	**0.00±0.00**%	**12.89±0.10**	19.42±0.12	10.62±0.57%	0.39±0.01
RL (w/ ACI)	94.08±1.18%	5.92±1.18%	**0.00±0.00**%	13.35±0.26	19.88±0.25	8.81±0.67%	0.40±0.00
Ours (w/ CV)	96.03±1.14%	3.73±1.24%	0.24±0.24%	17.88±0.60	24.51±0.76	2.40±0.22%	0.45±0.00
Ours (w/ GST)	**96.93±0.68**%	**2.93±0.61**%	0.13±0.12%	17.54±0.86	24.27±0.85	2.72±0.16%	0.44±0.00

TABLE II: Out-of-Distribution Test Results

Environments	Methods	**SR↑**	**CR↓**	**TR↓**	**NT↓**	**PL↓**	**ITR↓**	**SD↑**
20% Rushing Humans	SF	12.24%	19.12%	68.64%	32.06	36.15	5.31%	0.40
	ORCA	60.32%	34.96%	4.72%	23.41	19.84	**2.95** %	**0.48**
	MPC	49.76%	49.84%	0.40%	18.52	17.77	19.70%	0.39
	SafeCrowdNav	69.71±4.55%	30.29±4.55%	**0.00±0.00**%	13.47±0.40	**10.96±0.66**	20.20±1.10%	0.37±0.00
	CrowdNav++	73.17±1.24%	26.67±1.40%	0.16±0.16%	15.43±1.63	19.89±1.12	12.38±1.48	0.39±0.00
	RL (w/o ACI)	74.19±1.26%	25.81±1.26%	**0.00±0.00**%	**13.31±0.30**	18.20±0.10	18.23±0.31%	0.37±0.00
	RL (w/ ACI)	76.96±4.29%	23.04±4.29%	**0.00±0.00**%	14.13±0.32	19.04±0.14	15.84±1.14%	0.37±0.01
	Ours (w/ CV)	87.07±0.89%	**12.75±1.21**%	0.19±0.32%	18.74±0.31	24.57±0.66	5.29±0.20%	0.40±0.00
	Ours (w/ GST)	**87.17±4.14**%	**12.75±4.00**%	0.08±0.14%	18.32±1.01	24.04±0.85	6.82±1.36%	0.38±0.00
SF Pedestrian Model	SF	12.08%	6.72%	81.20%	29.76	40.56	1.60%	0.45
	ORCA	92.56%	4.88%	2.56%	22.36	21.91	**0.72**%	**0.48**
	MPC	89.76%	10.00%	0.24%	17.07	20.58	8.58%	0.41
	SafeCrowdNav	91.28±0.60%	8.72±0.60%	**0.00±0.00**%	**12.12±0.23**	**11.37±0.29**	10.94±0.40%	0.39±0.00
	CrowdNav++	92.48±1.36%	7.52±1.36%	**0.00±0.00**%	14.65±1.81	20.82±1.71	6.48±0.88%	0.41±0.00
	RL (w/o ACI)	95.68±0.89%	4.32±0.89%	**0.00±0.00**%	12.35±0.21	18.99±0.16	9.98±0.49%	0.39±0.00
	RL (w/ ACI)	97.41±0.81%	2.59±0.81%	**0.00±0.00**%	13.08±0.24	19.81±0.24	8.07±0.43%	0.40±0.01
	Ours (w/ CV)	98.48±0.92%	1.39±0.82%	0.13±0.12%	19.02±0.45	25.84±0.54	2.04±0.25%	0.43±0.01
	Ours (w/ GST)	**98.96±0.52**%	**1.04±0.52**%	**0.00±0.00**%	18.18±0.84	25.05±0.79	2.66±0.45%	0.42±0.00
Groups	SF	2.56%	97.44%	**0.00**%	**9.41**	**9.91**	23.59%	0.35
	ORCA	49.44%	46.72%	3.84	22.98	19.37	20.46%	0.39
	MPC	69.12%	29.76%	1.12%	20.48	22.78	24.08%	0.37
	SafeCrowdNav	75.73±2.01%	24.27±2.01%	**0.00±0.00**%	16.29±0.44	13.44±0.64	26.78±1.42%	0.35±0.01
	CrowdNav++	81.84±15.17%	18.13±15.20%	0.03±0.05%	17.28±1.30	24.95±3.60	9.05±6.54%	0.39±0.01
	RL (w/o ACI)	71.07±6.33%	28.93±6.33%	**0.00±0.00**%	14.87±0.67	21.35±1.08	20.17±2.07%	0.36±0.01
	RL (w/ ACI)	83.12±7.81%	16.88±7.81%	**0.00±0.00**%	15.70±0.39	23.53±1.30	12.67±4.83%	0.37±0.01
	Ours (w/ CV)	**94.51±4.20**%	**5.41±4.23**%	0.08±0.14%	21.16±1.32	30.61±1.98	**3.00±0.61**%	**0.43±0.01**
	Ours (w/ GST)	94.13±3.51%	5.71±3.31%	0.16±0.21%	20.19±0.69	29.69±0.81	3.55±1.21%	0.41±0.01

from the input and does not employ CRL, while RL (w/ ACI) incorporates uncertainty estimates but does not apply CRL.

C. In-Distribution Test Results

Quantitative Analysis. For all the test results in Table I-II, we evaluate 1250 samples across 5 random test seeds and calculate the mean performance and standard deviations of models trained with 3 different training seeds. The test results under the same setting as the training environment are shown in Table I. From these results, we can see that SF, ORCA, and MPC have lower SR and higher CR compared to RL methods. When comparing all the RL-based methods, both Ours (w/ GST) and Ours (w/ CV) outperform other methods in safety metrics and ITR, indicating that our method can generate both safe and polite trajectories that cause minimal intrusions to pedestrians. Although our method generates paths with higher TR, NT, and PL compared to RL baselines and ablation models, this tradeoff is reasonable considering the overall improvement in effectiveness and safety. Notably, Ours (w/ GST) achieves over an 8.80% higher SR, decreases collisions by more than 3.72 times, and reduces intrusions into the future pedestrian trajectories by more than 2.43 times compared to the two RL baselines.

Qualitative Analysis. We visualize the behaviors of Ours (w/ GST) and CrowdNav++ in the same episode in Fig. 2(a) and Fig. 2(b), respectively. At the beginning, CrowdNav++ decides to approach the goal directly. However, as the pedestrians move, they gradually surround the robot, which leads to an almost inevitable collision for CrowdNav++. In contrast, Ours (w/ GST) chooses to move the robot out of the crowds from the start. At step 24, Ours (w/ GST) rapidly reacts to a sudden change in human direction, and the expanding uncertainty area due to prediction errors accumulated helps the robot perform an avoidance maneuver.

D. Out-of-Distribution Test Results

OOD Scenarios Mixed with 20% Rushing Humans. In this setting, we set 20% of the human agents to have a maximum speed of $2.0\,\mathrm{m/s}$. From the results in Table II, we observe that all methods exhibit degraded performance compared to in-distribution conditions. Especially, SafeCrowdNav, CrowdNav++, RL (w/o ACI), and RL (w/ ACI) experience SR drops of 19.38%, 12.94%, 18.48%, and 17.12%, respectively. In contrast, Ours (w/ CV) and Ours (w/ GST) exhibit smaller SR drops of 8.96% and 9.76%, respectively. We visualize a scenario performed by Ours (w/ GST) in Fig. 2(c), where two

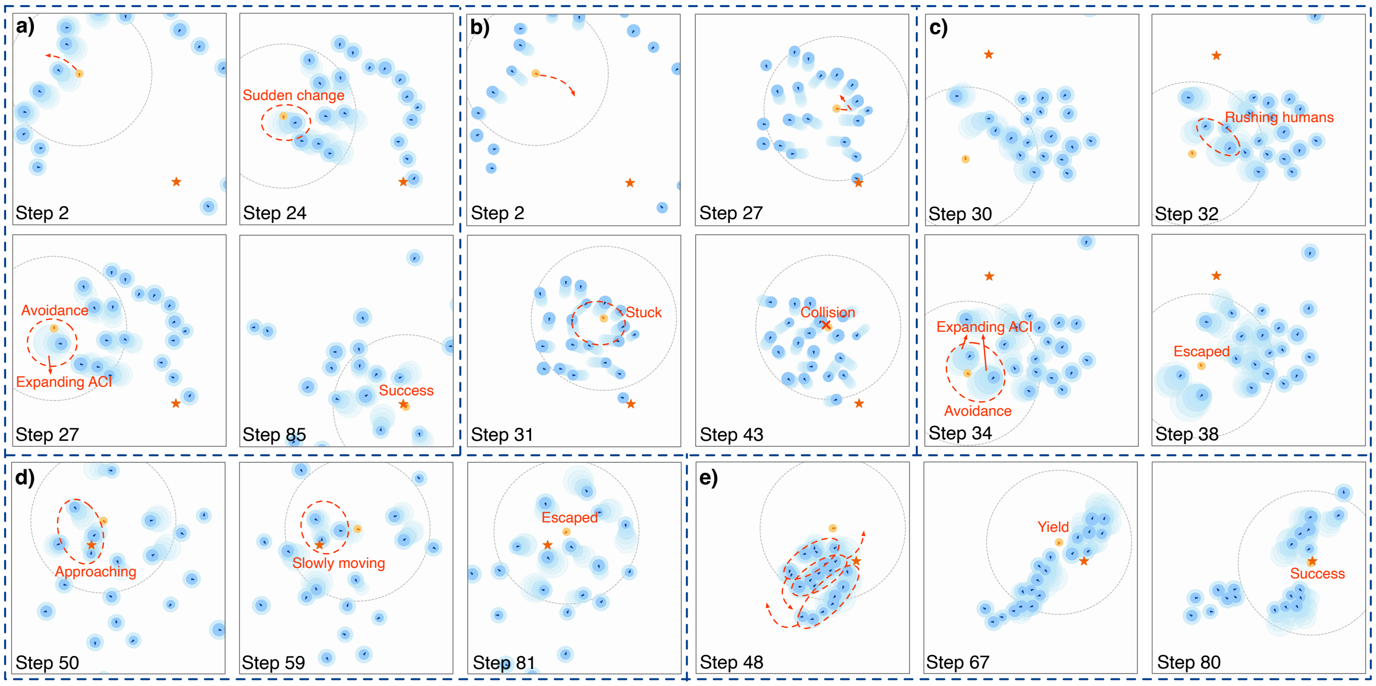

Fig. 2: Test-case visualizations. Pedestrians are shown in blue, the robot in yellow, and the goal as an orange star. Light-blue circles show the quantified uncertainties around pedestrians. (a) Ours (w/ GST) successfully navigates to the goal in an in-distribution environment. (b) CrowdNav++ fails to complete the same episode; here, light-blue circles represent predicted trajectories rather than uncertainties. (c) Ours (w/ GST) in an OOD environment with rushing pedestrians. (d) Ours (w/ GST) in an OOD environment using the SF pedestrian model. (e) Ours (w/ GST) in an OOD environment with pedestrian groups.

rushing humans are moving toward the agent at step 32. Due to the GST predictor facing the challenges of OOD conditions, the uncertainty area generated by ACI becomes much larger. The robot then chooses to navigate between these two agents through the gap between the uncertainty areas at step 34 and successfully escapes to a safe area at step 38.

OOD Scenarios with SF Pedestrian Model. In this setting, we change the behavior policy of all human agents from ORCA to SF. From the results in Table II, all methods perform better than in in-distribution conditions. Ours (w/ CV) and Ours (w/ GST) achieve almost perfect results in terms of SR, CR, and TR, implying that our methods *adapt well* to OOD scenarios caused by different behavior models. We visualize a scenario performed by Ours (w/ GST) in Fig. 2(d). When three humans approach each other, the robot with our policy maintains a conservative distance from the crowds and successfully escapes afterward.

OOD Scenarios with Group Dynamics. In this setting, pedestrians are clustered into cohesive groups that maintain tight intra-group spacing. As Table II shows, every RL baseline and ablation variant suffers a noticeable performance drop. Especially, SafeCrowdNav and RL (w/ ACI) fall to SR below 76%. The adaptability to grouped pedestrians of CrowdNav++ is highly sensitive to the training seed, with variance exceeding 15%. By contrast, Ours (w/ CV) and Ours (w/ GST) retain SR above 94% and CR below 6%, while also displaying the smallest variance across seeds.

E. Other Test Results

We include additional test results in Appendix E to H, covering the effectiveness of ACI quantification, a convergence comparison between Ours (w/ GST) and RL (w/ ACI), the influence of different cost limits on policy aggressiveness, and cross-evaluations of algorithms trained in environments where humans either actively avoid or ignore the robot.

F. Real-Robot Experiments

We deploy our method on a ROSMASTER X3 robot with Mecanum wheels running ROS2, connected via router to a laptop equipped with a mobile NVIDIA RTX 3070 GPU. We apply the model to the robot without further fine-tuning, using only basic clipping and smoothing. Perception is performed with a 6 Hz RPLIDAR-A1 LiDAR, human detection via a pre-trained DR-SPAAM model [18], tracking with SORT [2], and trajectory prediction with GST. We implement two movement modes: fixed-goal reaching and long-range navigation with a dynamically updated goal. The robot navigates outdoor environments and executes safe behaviors in both sparse and dense crowds. For more details, please refer to Appendix I-K and the supplementary video.

V. CONCLUSION

In this paper, we introduce an RL-based trajectory-planning framework that integrates conformal uncertainty into a CRL scheme to mitigate OOD performance degradation. Unlike conventional RL planners that overfit and falter under distribution shifts, our method dynamically leverages uncertainty estimates to adapt to velocity variations, policy changes, and transitions from individual to group dynamics. Extensive simulations demonstrate robust stability across diverse OOD scenarios, and real-world trials confirm the practical effectiveness of our approach.

References

[1] Anastasios N Angelopoulos, Stephen Bates, et al. Conformal prediction: A gentle introduction. *Foundations and Trends® in Machine Learning*, 16(4):494–591, 2023.

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

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

[4] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017.

[5] Anushri Dixit, Lars Lindemann, Skylar X Wei, Matthew Cleaveland, George J Pappas, and Joel W Burdick. Adaptive conformal prediction for motion planning among dynamic agents. In *Learning for Dynamics and Control Conference*, pages 300–314. PMLR, 2023.

[6] Anthony Francis, Claudia Pérez-d'Arpino, Chengshu Li, Fei Xia, Alexandre Alahi, Rachid Alami, Aniket Bera, Abhijat Biswas, Joydeep Biswas, Rohan Chandra, et al. Principles and guidelines for evaluating social robot navigation algorithms. *arXiv preprint arXiv:2306.16740*, 2023.

[7] Isaac Gibbs and Emmanuel Candes. Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 34:1660–1672, 2021.

[8] Isaac Gibbs and Emmanuel J Candès. Conformal inference for online prediction with arbitrary distribution shifts. *Journal of Machine Learning Research*, 25(162): 1–36, 2024.

[9] Harshayu Girase, Haiming Gang, Srikanth Malla, Jiachen Li, Akira Kanehara, Karttikeya Mangalam, and Chiho Choi. Loki: Long term and key intentions for trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9803–9812, 2021.

[10] Mahsa Golchoubian, Moojan Ghafurian, Kerstin Dautenhahn, and Nasser Lashgarian Azad. Uncertainty-aware drl for autonomous vehicle crowd navigation in shared space. *IEEE Transactions on Intelligent Vehicles*, 2024.

[11] James R Han, Hugues Thomas, Jian Zhang, Nicholas Rhinehart, and Timothy D Barfoot. Dr-mpc: Deep residual model predictive control for real-world social navigation. *IEEE Robotics and Automation Letters*, 2025.

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

[13] Huang Huang, Satvik Sharma, Antonio Loquercio, Anastasios Angelopoulos, Ken Goldberg, and Jitendra Malik. Conformal policy learning for sensorimotor control under distribution shifts. *arXiv preprint arXiv:2311.01457*, 2023.

[14] Zhe Huang, Ruohua Li, Kazuki Shin, and Katherine Driggs-Campbell. Learning sparse interaction graphs of partially detected pedestrians for trajectory prediction. *IEEE Robotics and Automation Letters*, 7(2):1198–1205, 2021.

[15] Zhe Huang, Tianchen Ji, Heling Zhang, Fatemeh Cheraghi Pouria, Katherine Driggs-Campbell, and Roy Dong. Interaction-aware conformal prediction for crowd navigation. *arXiv preprint arXiv:2502.06221*, 2025.

[16] Walter Jansma, Elia Trevisan, Álvaro Serra-Gómez, and Javier Alonso-Mora. Interaction-aware sampling-based mpc with learned local goal predictions. In *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 15–21. IEEE, 2023.

[17] Jiaming Ji, Jiayi Zhou, Borong Zhang, Juntao Dai, Xuehai Pan, Ruiyang Sun, Weidong Huang, Yiran Geng, Mickel Liu, and Yaodong Yang. Omnisafe: An infrastructure for accelerating safe reinforcement learning research. *arXiv preprint arXiv:2305.09304*, 2023.

[18] Dan Jia, Alexander Hermans, and Bastian Leibe. Dr-spaam: A spatial-attention and auto-regressive model for person detection in 2d range data. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10270–10277. IEEE, 2020.

[19] Bernard Lange, Jiachen Li, and Mykel J Kochenderfer. Scene informer: Anchor-based occlusion inference and trajectory prediction in partially observable environments. In *IEEE International Conferences on Robotics and Automation (ICRA)*, 2024.

[20] Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. *Advances in neural information processing systems*, 33, 2020.

[21] Jiachen Li, Xinwei Shi, Feiyu Chen, Jonathan Stroud, Zhishuai Zhang, Tian Lan, Junhua Mao, Jeonhyung Kang, Khaled S Refaat, Weilong Yang, et al. Pedestrian crossing action recognition and trajectory prediction with 3d human keypoints. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1463–1470. IEEE, 2023.

[22] Jiachen Li, Chuanbo Hua, Hengbo Ma, Jinkyoo Park, Victoria Dax, and Mykel J Kochenderfer. Multi-agent dynamic relational reasoning for social robot navigation. *arXiv preprint arXiv:2401.12275*, 2024.

[23] Kunming Li, Yijun Chen, Mao Shan, Jiachen Li, Stewart Worrall, and Eduardo Nebot. Game theory-based simultaneous prediction and planning for autonomous vehicle navigation in crowded environments. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2977–2984. IEEE, 2023.

[24] Lars Lindemann, Matthew Cleaveland, Gihyun Shim, and George J Pappas. Safe planning in dynamic environments using conformal prediction. *IEEE Robotics and Automa-*

tion Letters, 8(8):5116–5123, 2023.

[25] Shuijing Liu, Peixin Chang, Weihang Liang, Neeloy Chakraborty, and Katherine Driggs-Campbell. Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 3517–3524. IEEE, 2021.

[26] Shuijing Liu, Peixin Chang, Zhe Huang, Neeloy Chakraborty, Kaiwen Hong, Weihang Liang, D Livingston McPherson, Junyi Geng, and Katherine Driggs-Campbell. Intention aware robot crowd navigation with attention-based interaction graph. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12015–12021. IEEE, 2023.

[27] Licheng Luo and Mingyu Cai. Bridging deep reinforcement learning and motion planning for model-free navigation in cluttered environments. *arXiv preprint arXiv:2504.07283*, 2025.

[28] Christoforos Mavrogiannis, Francesca Baldini, Allan Wang, Dapeng Zhao, Pete Trautman, Aaron Steinfeld, and Jean Oh. Core challenges of social robot navigation: A survey. *ACM Transactions on Human-Robot Interaction*, 12(3):1–39, 2023.

[29] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 2019.

[30] 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*, 2024.

[31] Christoph Schöller, Vincent Aravantinos, Florian Lay, and Alois Knoll. What the constant velocity model can teach us about pedestrian motion prediction. *IEEE Robotics and Automation Letters*, 5(2):1696–1703, 2020.

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

[33] Kegan J Strawn, Nora Ayanian, and Lars Lindemann. Conformal predictive safety filter for rl controllers in dynamic environments. *IEEE Robotics and Automation Letters*, 2023.

[34] 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*, pages 1928–1935. Ieee, 2008.

[35] 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.

[36] Zhanteng Xie and Philip Dames. Scope: Stochastic cartographic occupancy prediction engine for uncertainty-aware dynamic navigation. *arXiv preprint arXiv:2407.00144*, 2024.

[37] Jing Xu, Wanruo Zhang, Jialun Cai, and Hong Liu. Safecrowdnav: safety evaluation of robot crowd navigation in complex scenes. *Frontiers in neurorobotics*, 17:1276519, 2023.

[38] Shuo Yang, George J Pappas, Rahul Mangharam, and Lars Lindemann. Safe perception-based control under stochastic sensor uncertainty using conformal prediction. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 6072–6078. IEEE, 2023.

[39] Rui Zhou, Hongyu Zhou, Huidong Gao, Masayoshi Tomizuka, Jiachen Li, and Zhuo Xu. Grouptron: Dynamic multi-scale graph convolutional networks for group-aware dense crowd trajectory forecasting. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 805–811. IEEE, 2022.

Appendix

A. *Preliminaries*

Adaptive Conformal Inference. Conformal methods can augment model predictions with a prediction set that is guaranteed to contain true values with a predefined coverage, enabling the quantification of uncertainties in a model-agnostic manner [8]. Traditional split conformal prediction requires a calibration set and places high demands on the exchangeability between the test sample and the calibration samples. In contrast, adaptive conformal inference (ACI) can dynamically adjust its parameters to maintain coverage in an online and distribution-free manner [7], making it appealing for time-sequential applications. Dynamically-tuned adaptive conformal inference (DtACI) [8] further boosts the applicability and performance of ACI by running multiple prediction error estimators with different learning rates simultaneously. DtACI adaptively selects the best output based on its historical performance, eliminating the need to pre-acquire underlying data dynamics to achieve satisfying coverage.

Constrained Reinforcement Learning. Constrained reinforcement learning (CRL) extends RL algorithms by incorporating constraints on the agent's behavior. Unlike traditional Markov decision process (MDP) settings, where agents learn behaviors only to maximize rewards, CRL is often formulated as a constrained Markov decision process (CMDP). At time step t, an agent chooses an action A_t under state S_t, receives a reward R_t, and incurs a cost C_t, after which the environment transitions to the next state S_{t+1}. In a CMDP, the objective is not only to find an optimal policy that maximizes rewards but also to manage costs associated with certain actions or states, which may be defined as quantities related to safety in the context of social navigation. This is generally represented as

$$\pi^* = \arg \max_{\pi \in \Pi^C} J^R(\pi), \tag{14}$$

where $J^R(\pi)$ is a reward-based objective function, and Π^C is the feasible set of policies that satisfy the constraints added to the problem. The goal of CRL is to ensure that costs remain within pre-defined thresholds while maximizing reward.

B. *Conceptual Differences with CBF*

In CBFs, a safety function $h(x)$ and a safe set $\{x : h(x) > 0\}$ are used to encode safety. The condition $\dot{h}(x) \geq -\gamma h(x)$ is enforced during optimization to ensure the system remains in the safe set, constraining the *derivative* of the safety function. In contrast, our framework focuses on controlling the *integral* of a risk-related function. Our cost function $C(S, A)$, analogous to $h(x)$, is accumulated over time as $\sum_{t=0}^{T} C_t(S_t, A_t)$, and we aim to constrain this cumulative cost below a small threshold $\tilde{d}_C$. Thus, while both approaches can shape the distribution of a safety-related function, the optimization mechanisms differ significantly. Our methods integrate more closely with CRL and achieve expected safety.

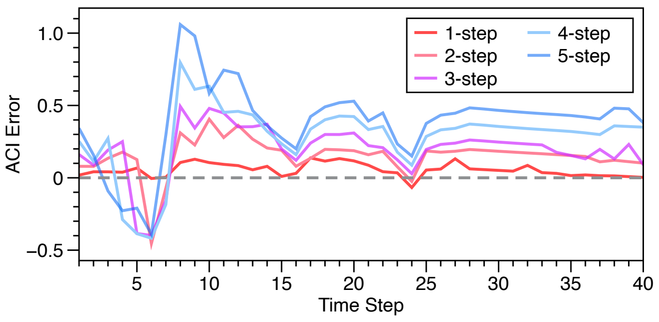

Fig. 3: Visualization of ACI errors (i.e., estimated prediction error minus actual prediction error) for one pedestrian's five prediction steps during the time period it is within the observable area of the robot. ACI provides valid coverage when the ACI error is greater than 0.

C. *Evaluation Metrics*

The evaluation metrics used in our paper are introduced in detail as follows:

- *Success Rate (SR):* SR measures the ratio of the number of successful episodes (i.e., the robot reaching the goal within the time limit of $50\,\text{s}$) to the total number of test episodes, i.e., SR $= N_{\text{success}}/N_{\text{total}}$.
- *Collision Rate (CR:* CR measures the ratio of episodes with at least one collision to the total number of test episodes, i.e., CR $= N_{\text{collision}}/N_{\text{total}}$.
- *Timeout Rate (TR):* TR measures the ratio of episodes that fail to reach the goal before the time limit to the total number of test episodes, i.e., TR $= N_{\text{timeout}}/N_{\text{total}}$.
- *Navigation Time (NT):* NT is the average time required for the robot to reach the goal (computed only over *successful* episodes), i.e., NT $= \left(1/N_{\text{success}}\right) \sum_{k=1}^{N_{\text{success}}} T_k$, where T_k is the time the robot took to reach the goal in the k-th successful episode.
- *Path Length (PL):* PL is the total distance traveled by the robot in an episode, accumulated step by step. We report the average path length across all episodes (including collisions and timeouts), i.e., PL $= \left(1/N_{\text{total}}\right) \sum_{i=1}^{N_{\text{total}}} \text{dist}_i$, where dist_i is the total distance traveled in episode i.
- *Intrusion Time Ratio (ITR):* ITR is the fraction of time steps in an episode where the robot intrudes into the ground-truth future positions of humans (*Danger* event triggered). Its final value is an average over all episodes:

$$\text{ITR} = \frac{1}{N_{\text{total}}} \sum_{i=1}^{N_{\text{total}}} \left(\frac{\#\text{ danger steps in episode } i}{\text{total steps in episode } i} \times 100\% \right). \tag{15}$$

- *Social Distance (SD):* SD is the average of the minimal distances between the robot and any human, computed only at the "danger steps." We log the minimal distance whenever a *Danger* event occurs, and then take the average over all episodes.

D. *Implementation Details*

Under each ablation setting, we train our models using three different random seeds on an NVIDIA RTX 4090 GPU, keeping all other hyperparameters consistent except those we intend to compare. Key common settings include:

- All models use human predictions as part of the input observations and employ a pre-trained GST predictor with fixed parameters to generate five steps of human trajectory predictions, except for Ours (w/ CV).
- For the training parameters, we leverage the parallel training feature of PPO by setting the batch size to 32 and running 128 parallel environments. The learning rate is set to 3×10^{-5}, and the clip parameter is set to 0.08 for both the actor and the reward critic. The learning rate for the cost critic is set to 1.5×10^{-5}. We run 2×10^{7} training steps to obtain the final policy model for testing and real-robot deployment. In our training, varying entropy is removed, and the action noise, which is normally distributed, is only added during training and removed during testing and deployment.
- We set the radius of the cost subareas around current positions of humans, r_{comfort}, to $0.25\,\text{m}$ and calculate costs based on the intrusions into the subarea and the first two prediction uncertainty areas, while inputting predictions for all five future steps into the policy network with a coverage parameter $\alpha = 0.1$, corresponding to 90% coverage in prediction errors.
- For DtACI hyperparameters, the initial prediction errors are set to $0.1\,\text{m}$, $0.2\,\text{m}$, $0.3\,\text{m}$, $0.4\,\text{m}$, and $0.5\,\text{m}$ for 1- to 5-step-ahead predictions, respectively. We employ three error estimators with learning rates γ of 0.05, 0.1, and 0.2 for each DtACI estimator.
- For the specific reward and cost functions, we set the success reward R_{success} to $+10$, the collision penalty $R_{\text{collision}}$ to -20, and the dense potential reward that guides the robot's movement to $2\Delta d_{\text{forward}}$, where $\Delta d_{\text{forward}}$ measures the distance the robot moves toward the goal compared to the previous step. The cost $C_t(S_t, A_t)$ is defined as $2.5 d_{\text{intru},t}$, where $d_{\text{intru},t}$ represents the maximum intrusion into the cost areas of the current human positions and the first two prediction uncertainty areas.

E. ACI Effectiveness

To validate the effectiveness of ACI in covering actual prediction errors, we visualize the ACI errors of one human in Fig. 3. At the beginning of this trajectory, the ACI errors for multi-step predictions are large because the GST predictor lacks sufficient information to accurately predict human positions, leading to large actual prediction errors that our initial ACI values do not cover properly. However, after several steps, ACI quickly adapts to the actual prediction error and achieves adequate coverage. Additionally, if the coverage remains sufficient, the ACI error will decrease to ensure that the uncertainty estimation is not too conservative.

F. Convergence Analysis

We present the learning curves of Ours (w/ GST) in Fig. 4(a), alongside RL (w/ ACI) for comparison. The results indicate that Ours (w/ GST) demonstrates a smoother learning process and achieves higher rewards. Furthermore, as shown in Fig. 4(b), the average episodic costs of Ours (w/ GST) converge to different cost limit values. This provides an effective mechanism for adjusting the aggressiveness of robot policies, as further discussed in Appendix G.

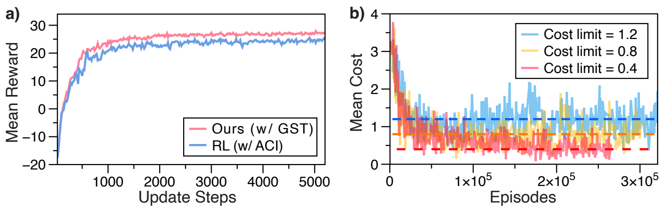

Fig. 4: Convergence analysis of our method. (a) The learning curves of Ours (w/ GST) and RL (w/ACI). Ours (w/ GST) shows faster convergence with higher rewards. (b) The cost curves of Ours (w/ GST) with different cost limits. The average costs across episodes can approximately approach the predefined cost limits, which are shown by the dashed lines.

TABLE III: Performance of Ours (w/ GST) Under Different Cost Limits

Cost limit	SR↑	CR↓	TR↓	NT↓	PL↓	ITR↓	SD↑
0.4	96.96%	2.80%	0.24%	18.52	25.24	2.73%	0.43
0.6	96.72%	3.04%	0.24%	16.16	22.96	3.59%	0.44
0.8	94.56%	5.44%	0.00%	13.60	20.33	5.61%	0.43
1.0	94.32%	5.68%	0.00%	13.53	20.19	6.62%	0.42
1.2	93.76%	6.24%	0.00%	12.97	19.45	7.31%	0.41

G. Tuning Aggressiveness of Trajectories with Cost Limits

Although the main results presented in our paper focus on low cost limits to ensure safe navigation through crowds, we also observe interesting testing results with varying cost limits that allow for tuning the aggressiveness of trajectories, as shown in Table III. These results are obtained by testing across five seeds, with 250 test samples for each seed, using models trained with the same training seed. We find that as the cost limits increase, the trajectories become gradually more aggressive, reflected by an increase in ITR. This increased aggressiveness also improves efficiency, as indicated by reductions in NT and PL. However, it comes at the expense of key metrics such as SR and CR. Therefore, for a single best policy, we recommend keeping the cost limit as low as possible, provided the policy can still converge.

H. Visible Robot Settings

For the main results presented in our paper, we train the RL baselines, ablation models, and our methods in the CrowdNav environments where humans will not actively avoid the robot, which is a common practice in prior work [26, 37]. To validate the effectiveness of this setting, we conduct a cross-evaluation on models trained in environments where humans either actively avoid or ignore the robot, as summarized in Tables IV-V. Specifically, we evaluate the models under five different testing seeds, each with 250 test samples, for three models trained under three distinct training seeds. From these results, we observe that although models trained in the visible robot setting achieve low collision rates in their in-distribution settings, they fail to generalize when humans do not react to the robot. In such scenarios, nearly half of the test cases result in collisions. In contrast, models trained in the invisible robot setting maintain low collision rates in both testing scenarios

TABLE IV: Cross-Validation of Ours (w/ GST) for Robot Visibility Settings Using Success Rates

Test \ Train	Invisible	Visible
Invisible	**96.93±0.68**%	50.48±0.49%
Visible	**99.92±0.14**%	99.07±0.76%

TABLE V: Cross-Validation of Ours (w/ GST) for Robot Visibility Settings Using Collision Rate

Test \ Train	Invisible	Visible
Invisible	**2.93±0.61**%	49.52±0.49%
Visible	**0.08±0.14**%	0.93±0.76%

and, surprisingly, exhibit even lower collision rates when the robot is visible than those trained in the visible robot setting. In real-world situations, some pedestrians may ignore the robot (e.g., when they are using their mobile phones while walking). This mixture of reactive and non-reactive pedestrians poses significant challenges for models that assume all humans will actively avoid the robot. Thus, training in an environment where humans do not react to the robot proves to be more robust, as it better accounts for scenarios in which pedestrians may fail to notice the robot.

I. ROS2 System for Real Robot Deployment

We develop a full ROS2 system from perception to decision making, including four main nodes:

- *Detector:* This node employs a pretrained DR-SPAAM model [18] to detect human agents from 2D LiDAR point clouds. The model leverages a lightweight neural network, enabling real-time detection on resource-constrained devices. According to the original paper, it achieves an accuracy metric of $AP_{0.5} = 70.3\%$.
- *Tracker:* This node uses the SORT [2] algorithm for tracking, which simply aims at assigning indices to detected agents. To generate consistent tracking results even when receiving noisy detections, we assign large values to the measurement noise covariance matrix $\mathbf{R}$ and the process noise covariance matrix $\mathbf{Q}$ of the underlying Kalman filter to accommodate detection jitter. We also set a higher noise coefficient for the position and scale components of the Kalman filter and a smaller value for the velocity components to allow more flexibility in position and scale updates. In the initial state covariance matrix $\mathbf{P}$, the velocity components are given a large uncertainty, and the entire matrix is scaled accordingly so the filter can quickly adapt during early tracking. Besides, in the SORT tracker, we only keep trajectories whose ratio of valid detections over the entire detected period is above 0.9, filtering out spurious or short-lived detections. At each timestep, we run the Kalman filter's inference step to infer the positions of tracked agents even when they are not correctly detected in certain frames.
- *Predictor:* This node uses the GST [14] model as the trajectory planner, integrating the DtACI module to directly apply uncertainty quantification after obtaining prediction lines for the pedestrians. The parameters and model weights are consistent with those used during training.
- *Decider:* This node receives information from the upper-level nodes and determines the output action commands (v_x and v_y for velocities) for the controller. We utilize the controller integrated into the ROSMASTER X3 system by publishing `\cmd_vel`. The decider supports three modes: goal-reaching mode, long-range navigation mode, and manual control mode. The selection of these modes is managed by another utility node, *Command Listener*, which listens to user input and communicates the commands to the decider.

J. Real Robot Experiments

We deploy our approach on a ROSMASTER X3 robot equipped with Mecanum wheels, enabling flexible movement and independent control of v_x and v_y. The robot connects to a laptop with an Nvidia RTX 3070 GPU (mobile version) via a router. Our model is trained in CrowdNav simulations with holonomic dynamics and is deployed directly to the robot system *without* further fine-tuning.

The robot uses a 2D LiDAR (RPLIDAR-A1) for human detection via a learning-based detector. The LiDAR operates at a scanning frequency of approximately $6\,\mathrm{Hz}$, which also limits the frequencies of the tracking, prediction, and decision making modules. Despite this, our experiments demonstrate that the robot makes effective navigation decisions in crowded environments, indicating that the system generalizes well to cost-efficient robots. Human detection is done with a pre-trained DR-SPAAM model [18], tracking is done with SORT [2], prediction is done with the GST model [14], and decision making is powered by Ours (w/GST). For further details about our system, please refer to Appendix I, and computational speed analyses for the core functions of each module are provided in Appendix K.

We conduct the experiments on a large outdoor terrace covering an area greater than $15\,\mathrm{m} \times 20\,\mathrm{m}$. The robot operates in two movement modes:

- *Goal-reaching mode:* The robot navigates to a predetermined target point, which is the original setting in the standard CrowdNav training phase.
- *Long-range navigation mode:* The goal of the robot is dynamically updated based on its movement. This approach addresses the limitation that, during training, the goal inputs to the policy networks are constrained to a limited range. As a result, distant goals beyond $12\,\mathrm{m}$ may negatively impact policy performance. By introducing a continuously updated moving target, we show that the robot can reliably navigate distances exceeding $20\,\mathrm{m}$ through experiments. This method has the potential to be extended to even longer distances if the system is fully integrated onboard in the future.

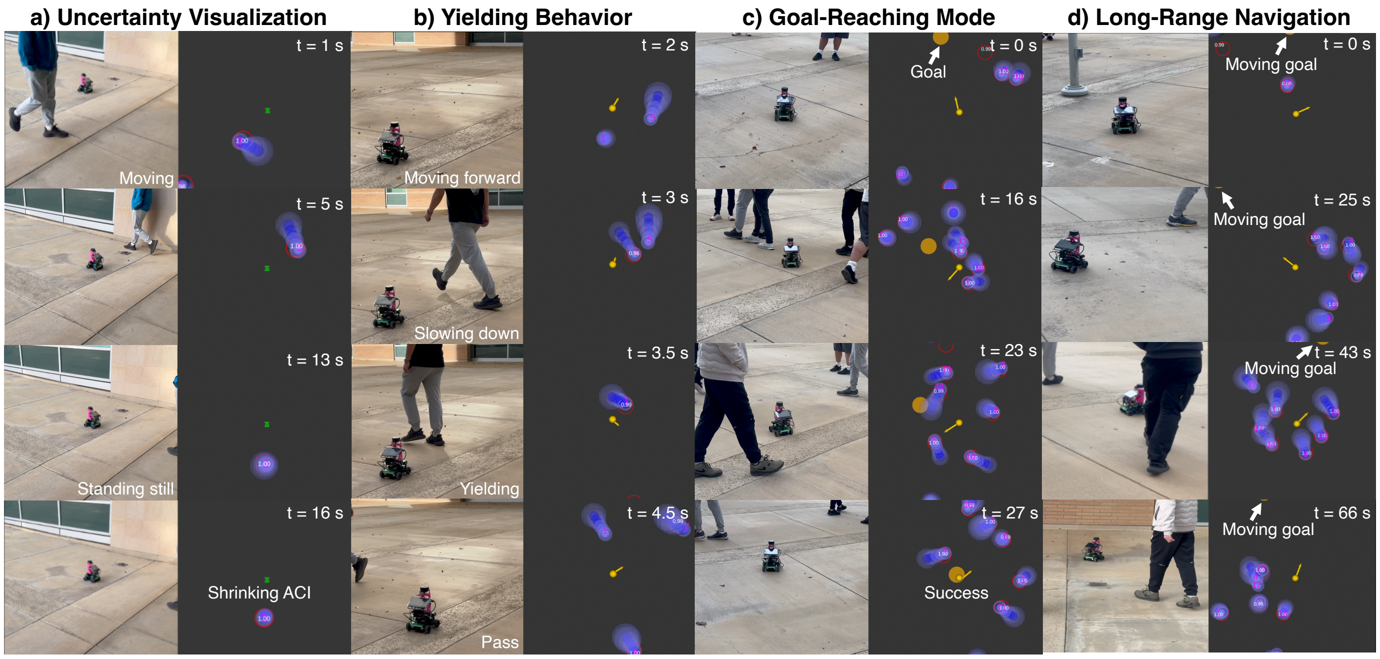

Fig. 5: We deploy our methods on a ROSMASTER X3 with Mecanum wheels using the ROS2 system. For the four subplots, the left sides display photos taken from the experiments, and the right sides show visualizations in RViz. In the RViz visualizations, the red circles represent detection results, and the white numbers inside the red circles indicate the output probabilities of the detection model. The purple numbers correspond to the indices generated by the tracker. The prediction lines are shown in blue, and the prediction uncertainties are depicted in semi-transparent light blue. In (a), the green robot represents the robot's current location. In (b)-(d), where the decision node is enabled, the yellow sphere indicates the robot's position, and the yellow arrow represents the command output from the decider node. The orange circle indicates the goal position. (a) In the uncertainty visualization, the human initially walks around the robot and then stands still behind it. The uncertainty area adjusts dynamically based on the prediction accuracy. (b) The robot equipped with Ours (w/ GST) demonstrates stable yielding behavior when interacting with humans. (c) In goal-reaching mode, the robot navigates through crowds and successfully reaches its goal. (d) In long-range navigation mode, the moving goal consistently guides the robot's movement.

We show several representative real-world testing scenarios and results in Fig. 5. Specifically, we first visualize the prediction uncertainties using RViz in Fig. 5(a). Before $t = 13\,\mathrm{s}$, the human walks around the robot, and the robot successfully detects the human's position while visualizing the predicted trajectory and the associated uncertainty area. Since the prediction model does not fully capture the human's circular walking pattern, the long-horizon prediction uncertainty area is significantly larger than that of short-horizon predictions. After $t = 13\,\mathrm{s}$, the human stands still behind the robot. Initially, the prediction uncertainty area is large because the uncertainty results are carried over from iterative processes. Gradually, the uncertainty area shrinks to reflect more precise prediction uncertainties. After approximately 3 seconds, the uncertainty area stabilizes and reduces to almost zero. This process demonstrates that by using DtACI, the system can effectively adapt to different conditions and dynamically reflect prediction uncertainties. This enables safer decision making without being overly conservative.

In Fig. 5(b), by allowing the robot to interact with a single agent, we observe that the robot can generate stable yielding behaviors. At $t = 2\,\mathrm{s}$, the robot detects that a human is coming along a path that will intersect with its original moving trajectory. It begins by slowing down to reduce the risk of a collision ($t = 3\,\mathrm{s}$). At $t = 3.5\,\mathrm{s}$, the robot adjusts its direction, moving towards the backside area of the human to avoid intrusions. At $t = 4.5\,\mathrm{s}$, the robot successfully passes.

In Fig. 5(c), the robot navigates in goal-reaching mode while interacting with humans along its path. We observe that the robot encounters very dense crowds at $t = 16\,\mathrm{s}$ and $t = 23\,\mathrm{s}$. Throughout this process, the robot makes appropriate decisions to minimize intrusions into humans' trajectories while progressing toward its goal, and it ultimately reaches the goal despite the dense human interactions.

In Fig. 5(d), we implement the long-range navigation mode, enabling the robot to navigate distances beyond those encountered during training. In this mode, the moving goal is dynamically updated to remain $5\,\mathrm{m}$ ahead of the robot's position along the longitudinal axis, while its lateral position remains fixed. Under these conditions, the robot moves forward while actively avoiding dynamic obstacles, as demonstrated at $t = 25\,\mathrm{s}$. Although the robot temporarily deviates laterally to avoid collisions, the fixed lateral component of the moving goal serves as a stable reference, helping the robot stay within a reasonable lateral range.

Overall, our system demonstrates robust decision-making capabilities in crowded environments. By quantifying prediction uncertainty and incorporating it into behavior guidance, the robot develops more adaptive spatial awareness and exhibits consistent, safe, and socially aware behaviors.

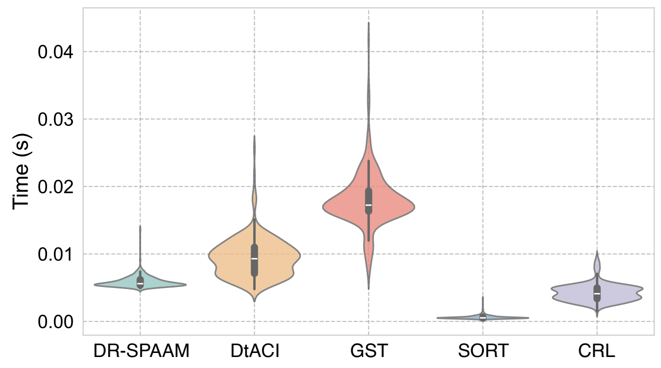

Fig. 6: Visualizations of computational time distributions for five key modules of our ROS2 system. The data samples were collected during experiments in dense human environments. The shape of each violin plot represents the density of computation times, with wider areas indicating a higher concentration of values. The white dots at the center of each shape denote the mean computation time for the respective module.

K. Computational Speed Analysis

We log the time consumption of key functions across different nodes, including the inference times of the detection model (DR-SPAAM), prediction model (GST), and the policy network (CRL), as well as the update times for the tracker (SORT) and uncertainty quantifier (DtACI) in dense crowd experiments, and visualize the results in Fig. 6. The visualization shows that GST has the longest average computation time and the largest variance. In contrast, DR-SPAAM and SORT maintain low and stable computation times, making them efficient for real-time applications. While DtACI has relatively high computational costs due to its pure Python implementation, its average time remains below $0.01\,\mathrm{s}$, well within acceptable limits. This overhead can be greatly reduced by transitioning to more efficient languages like C++. Despite this, the worst-case computation time for all main modules stays below 0.1 seconds, meeting and exceeding our system's decision frequency requirements.