A Closed-Loop Perception, Decision-Making and Reasoning Mechanism for Human-Like Navigation

Wenqi Zhang[1*] , **Kai Zhao**[2*] , **Peng Li**[3,6] , **Xiao Zhu**[4] ,
Yongliang Shen[1] , **Yanna Ma**[5] , **Yingfeng Chen**[2] and **Weiming Lu**[1†]
[1]College of Computer Science and Technology, Zhejiang University
[2]Netease Fuxi Robot Department
[3]Institute of Software, Chinese Academy of Sciences
[4]College of Mechanical Engineering, Zhejiang University of Technology
[5]University of Shanghai for Science and Technology
[6]University of Chinese Academy of Sciences Nanjing
{zhangwenqi, luwm}@zju.edu.cn,{zhaokai02,chenyingfeng1}@corp.netease.com, lipeng@iscas.ac.cn

Abstract

Reliable navigation systems have a wide range of applications in robotics and autonomous driving. Current approaches employ an open-loop process that converts sensor inputs directly into actions. However, these open-loop schemes are challenging to handle complex and dynamic real-world scenarios due to their poor generalization. Imitating human navigation, we add a reasoning process to convert actions back to internal latent states, forming a two-stage closed loop of perception, decision-making, and reasoning. Firstly, VAE-Enhanced Demonstration Learning endows the model with the understanding of basic navigation rules. Then, two dual processes in RL-Enhanced Interaction Learning generate reward feedback for each other and collectively enhance obstacle avoidance capability. The reasoning model can substantially promote generalization and robustness, and facilitate the deployment of the algorithm to real-world robots without elaborate transfers. Experiments show our method is more adaptable to novel scenarios compared with state-of-the-art approaches.

1 Introduction

Safe, reliable, and flexible navigation and obstacle avoidance strategies are essential for large-scale robotic and autonomous driving applications. In a complex scenario of human-robot coexistence, the agent needs to avoid highly dynamic obstacles and drive quickly to the target, which is fairly challenging. Most conventional algorithms[Fox *et al.*, 1997] implement navigation through real-time path planning, which requires a high computational overhead.

Recent advances in deep learning have greatly improved perception capabilities and even surpassed human levels in many areas[He *et al.*, 2016b; Vaswani *et al.*, 2017]. But supervised learning paradigm, which relies on large amounts of training data, does not easily tackle complex navigation problems in highly dynamic scenarios. Navigation is inherently a complex sequential decision-making task, including global path planning and local obstacle avoidance. It is impractical to collect sufficient data for training. Obviously, Complex decision-making tasks in navigation and obstacle avoidance cannot be well solved by perception model alone.

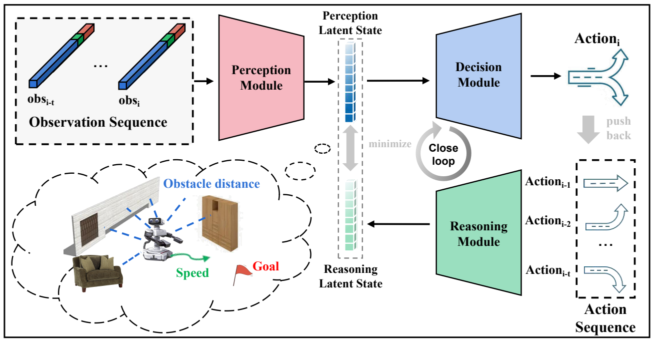


Figure 1: We propose a closed-loop mechanism including perception, decision-making, and reasoning model. The perception model extracts a latent representation of both spatial and temporal aspects of observation. The decision-making model calculates the most appropriate action according to the latent state. The reasoning model imagines this latent representation of the surroundings through the action sequences in memory.

Deep reinforcement learning(DRL)[Mnih *et al.*, 2013; Sutton and Barto, 2018] has been widely studied and applied in many fields, including robot control[Yang *et al.*, 2020; Fu *et al.*, 2021], autonomous driving[Long *et al.*, 2018], etc. The DRL-based approach shows great potential to build a reliable decision-making system. Currently, many researchers have applied DRL to navigation and obstacle avoidance[Faust *et al.*, 2018; Zhang *et al.*, 2021], achieving promising performance. Most of these works attempt to learn a precise policy from observations to actions during environment interactions. However, such one-way learning process is open-loop and lacks a deep understanding of the action. These open-loop

*Contributed equally to this work. †Corresponding author.

methods perform well in trained scenarios, but may not be robust in general scenarios without understanding the intrinsic relations between internal states and output action. Besides, DRL-based methods are fragile and unstable when deployed to real scenarios, since the real world is dynamic, complex, and always changing.

However, for the person with visual impairment, we observe a reasoning process in their mind. They perceive the outside obstacles through their crutch and imagine the surroundings based on their motion when exploring an unfamiliar scene. The imaginary scenes constructed in their minds match the real scenes in reality, forming a closed loop. Inspired by this, we design a two-stage closed-loop mechanism with perception, decision-making, and reasoning components. As shown in Figure 1, the perception model is to compress observation into latent state distribution and the decision-making model converts the latent state into current action, just as a blind person uses a crutch to explore a new scene. Simultaneously the reasoning model deduces the most likely latent states based on the action sequences, like the person imagining the surrounding scene in their mind. This reasoning process constructs a closed-loop learning process by parsing the intrinsic relation between actions and the latent states, generating more reasonable actions than the open-loop system. Unlike conventional feedback control algorithms, our algorithm focuses on the closed loop of the learning process, eliminating the need to accurately model the kinematics and dynamics of the robot and the environment.

We assess our algorithm on two benchmarks that we design for evaluating navigation ability in few-shot and zero-shot scenes. Experiments demonstrate that our approach achieves significant improvement over various baselines, and is more reliable in novel scenarios. In addition, we deploy the algorithm to a real robot in a crowded building. Despite the significant gap between the simulator and the real world, the agent achieves autonomous obstacle avoidance without any human assistance. The main contributions of this work can be summarized as follows:

- We propose a perception, reasoning, and decision-making mechanism, which can learn more general rules and robust strategies through the reasoning model and latent state distribution.
- We introduce VAE-Enhanced Demonstration Learning to acquire basic navigation rules from data and design two dual-learning processes in RL-Enhanced Interaction Learning to promote collision avoidance ability by generating reward feedback for each other.
- Extensive experiments show our method achieves the most reliable results, surpassing the previous approaches in terms of safety and task accomplishment.

2 Related Work

2.1 Navigation and Planning

In recent years, autonomous navigation and flexible obstacle avoidance have attracted extensive attention. Conventional navigation approaches[Khatib, 1986] are usually implemented by global and local path planning[Fox *et al.*, 1997]. These approaches have trouble addressing navigation challenge in an unknown environment as they must be provided with the map in advance. In addition, many dynamic path planning and re-planning algorithms[Stentz and others, 1995; Koenig and Likhachev, 2002; Qi *et al.*, 2021] can also achieve path planning in unknown dynamic environments, but the high computational overhead of these algorithms constrains real-time performance in large-scale dynamic environments. Besides, most of them require fine-tuning of parameter in real applications.

With the breakthrough of deep learning, many researchers have deployed supervised learning and reinforcement learning to achieve automatic obstacle avoidance[Chen *et al.*, 2017]. But collecting abundant samples for training is another annoying issue, as manual annotation requires tremendous tedious labor. To overcome this limitation, some researchers attempt to adopt imitation learning[Ho and Ermon, 2016; Pfeiffer *et al.*, 2018]. Imitation algorithm designs a teacher-student learning paradigm to mimic the teacher's action output and the intention behind it. Furthermore, some researches focus on modeling the environment. [Ha and Schmidhuber, 2018] adopted a variational autoencoder(VAE) to learn a compressed spatial and temporal representation of the environment. Although the above-mentioned studies have strikingly improved the performance of the navigation, most of them are based on learning an explicit mapping from perfect observations to deterministic actions. Such an assumption is impractical in a real scenario. These methods may have weak robustness and generalization, especially when confronted with novel scenarios.

2.2 Deep Reinforcement Learning

Deep Reinforcement Learning(DRL) is an optimization algorithm based on Markov Decision Process(MDP). Its optimization goal is to maximize the expected accumulated discount rewards[Sutton and Barto, 2018; Mnih *et al.*, 2013]. Many researchers have applied DRL to traditional decision-making tasks and achieved remarkable results. [Tsounis *et al.*, 2020] applied DRL to robot control and realized autonomous walking of a quadruped robot. [Long *et al.*, 2018; Zhang *et al.*, 2021] introduced DRL to solve path planning and obstacle avoidance in a dynamic environment. Some low-resource tasks[He *et al.*, 2016a; Artetxe *et al.*, 2017; Cao *et al.*, 2020] can also be solved using dual DRL. These dual-DRL frameworks utilize the dual property of two tasks to enhance the model performance. However, DRL also has some inherent drawbacks that hinder its performance. One of the troubling challenges is commonly referred to as the "curse of dimensionality", which hinders the collision avoidance performance in large-scale real scenarios. The other is the poor generalizability and robustness when encountering complex and dynamic scene.

3 Approach

In this section, we first introduce the perception, reasoning, and decision-making components, and then describe in detail the design of VAE-Enhanced Demonstration Learning and RL-Enhanced Interaction Learning.

3.1 Model Definition

In previous approaches, the model directly maps external observations into actions [Long *et al.*, 2018; Zhang *et al.*, 2021]. In fact, these observations also contain lots of irrelevant information and measurement noise. We use o_t to denote the current observation, a_t for current action, and s_t for current latent state based on observation. Assuming that s_t obeys a normal distribution $N_t(s)$ and it represents the valid features extracted from o_t. The perception, decision-making and reasoning models achieves the transition between o_t, a_t, s_t. The perception and reasoning are two sequence encoders based on observations and actions, while decision-making is a simple decoder depending on the current latent state. This process is similar to human navigation that the perception and reasoning are based on long-term memory, while decision making is an instant subconscious response.

Perception component. The perception model is a sequence encoder P that maps observations sequences $o_{t-n:t}$ into the latent state distribution N_t^P, where $o_{t-n:t} = o_{t-n} \cdots o_t$ denotes the sequences of observation, and latent state is sampled from latent state distribution, i.e. $s_t^P \sim N_t^P$. The perception model extracts useful features from the observed sequences, and compresses them into the latent state, filtering out the irrelevant information.

Decision-Making component. The decision model is a direct mapping from latent states to actions, i.e. $a_t = D\left(s_t\right)$.

Reasoning component. The reasoning model R is to deduce the latent state distribution N_t^R based on the action sequence $a_{t-n:t}$, where $a_{t-n:t} = a_{t-n} \cdots a_t$ represent the sequences of actions. Another latent state is sampled from this distribution, i.e. $s_t^R \sim N_t^R$. This reasoning process is similar to a person imagining the surrounding scenes based on movements.

3.2 VAE-Enhanced Demonstration Learning

Using DRL to directly train navigation policy from scratch is inefficient since DRL usually encounters a cold start problem. We design a VAE-Enhanced phase to learn from data. This process is similar to imitation learning, where three models learn the general rules from the data. The overview of this process is visualized in Figure 2. It contains two channels:

- *Prediction-Denoise-channel*: decision-making (D) and perception model(P)
- *Reconstruction-VAE-channel*: decision-making(D) and reasoning model(R)

Prediction-Denoise-channel. Pair data$\{o_t, a_t\}_{t=1:N}$ sampled from collected dataset (See section 4.1 for more detailed descriptions about dataset). It means that when o_t is observed, the agent should take a_t. $p(s_t|o_{t-n:t})$ means perception model and $p(a_t|s_t)$ represents decision-making model. Latent state cannot be observed. The likelihood as follows:

$$p(a|o) = \int p(s,a|o)ds = \int p(a|s)p(s|o)ds = \int D(\cdot)P(\cdot)ds \tag{1}$$

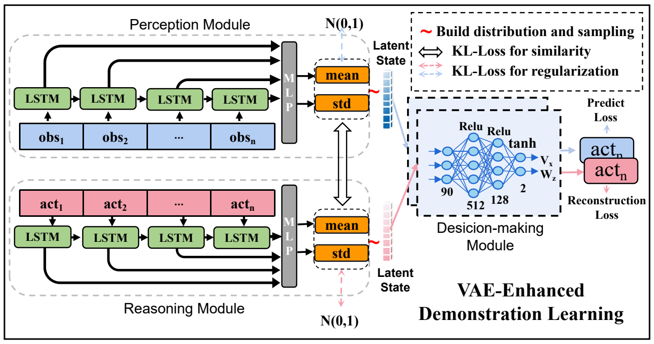


Figure 2: Overview of the VAE-Enhanced Demonstration Learning.

The algorithm maximizes the log-likelihood of the data $\{o_t, a_t\}$. So, we adopt a *Prediction-Denoise-channel* to achieve this objective. The perception model converts observation sequences $o_{t-n:t}$ into latent distribution $N_t^P(s)$, and then the decision-making model converts latent state with sampling noise s_t^P to action a_t. The perception and decision-making models are more robust for predicting actions due to the presence of sampling noise in latent state. The *prediction-channel* can be formalized as

$$\mu_t^P, \sigma_t^P = \text{MLP}^P(\text{LSTM}^P(o_{t-n:t})) \tag{2}$$

$$s_t^P \sim N(\mu_t^P, \sigma_t^P) \tag{3}$$

$$a_t^{pred} = \text{MLP}^D(s_t^P) \tag{4}$$

Reconstruction-VAE-channel. Inspired by [Kingma and Welling, 2014], we adopt a *Reconstruction-VAE-channel* to extract a continuous mapping for reasoning model. Assume that true distribution from action to latent state is $p^*(s|a)$, and the reasoning model $R(s_t|a_{t-n:t})$ achieve an estimate of this distribution. To minimize the KL divergence of the two distributions:

$$\begin{aligned} KL\left(R\left(s|A\right)\|p^*(s|a)\right) &= \int R(s|A)\log\frac{R(s|A)}{p^*(s|a)}ds \\ &= \int R(s|A)\log\frac{R(s|A)}{p^*(s)}ds - \int R(s|A)\log p(a|s)ds \\ &= KL(R(s|A)\|p^*(s)) - E_{s\sim R(s|A)}[\log p(a|s)] \\ &= KL(R(\cdot)\|p^*(s)) - E_{s\sim R(\cdot)}[\log D(\cdot)] \end{aligned} \tag{5}$$

Where A means action sequence. The reasoning and decision-making models constitute a standard VAE process, which achieves the reconstruction of the action. Given an action sequence, reasoning model encodes $a_{t-n:t}$ into latent state distribution N_t^R, i.e. $\mu_t^R, \sigma_t^R = \text{MLP}^R(\text{LSTM}^R(a_{t-n:t}))$. Decision-making model samples another latent state s_t^R, and then reconstructs action a_t, denoted as $a_t^{reconst} = \text{MLP}^D(s_t^R) = \text{MLP}^D(s \sim N_t^R)$.

Two channels share one decision-making model, achieving action prediction and reconstruction, respectively. The training objective can be formulated as

$$L_{1,t} = (a_t^{reconst} - a_t)^2 + (a_t^{pred} - a_t)^2 \tag{6}$$

$$L_{2,t} = \text{JS}(N(\mu_t^P, \sigma_t^P)\|N(\mu_t^R, \sigma_t^R)) \tag{7}$$

$$L_{3,t} = \sum_{\Delta\in\{P,R\}} \frac{1}{2}(-1 - \log\sigma_t^\Delta + (\mu_t^\Delta)^2 + \sigma_t^\Delta) \tag{8}$$

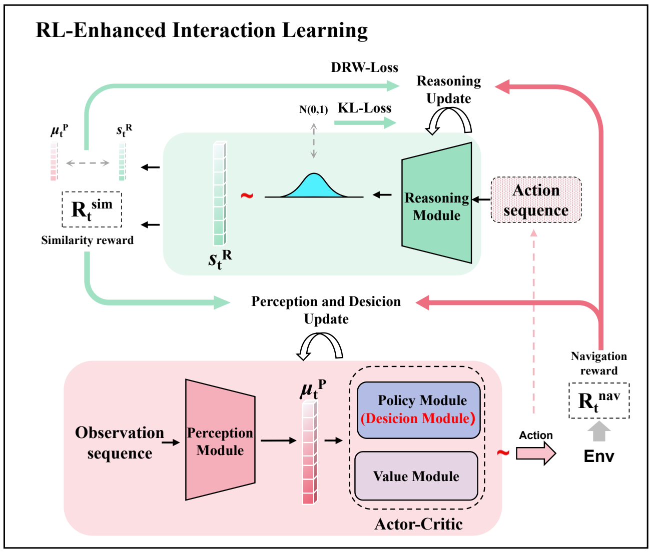


Figure 3: The reasoning model updates itself at a fixed interval by selecting high navigation-reward actions. Meanwhile, the reasoning model generates a similarity-reward to evaluate each action generated by the decision-making model during the interaction.

The total loss is $L = \sum_t(L_{1,t} + L_{2,t} + L_{3,t})$, where $t = 1, 2 \cdots K$, K is the total number of samples and JS stands for Jensen–Shannon divergence. L_1 represents the prediction loss and reconstruction loss for two channels. L_2 constrains that the latent state distribution calculated by the perception and reasoning models should be similar. L_3 is the regularization constraints two latent distributions to the standard normal distribution. The introduction of the VAE and two channels enhance the generalization and robustness of the three models, avoiding overfitting sparse data.

3.3 RL-Enhanced Interaction Learning

After VAE-Enhanced Demonstration Learning, the model is able to learn general collision avoidance rules. However, the limited training data still confines the performance of the algorithm in highly dynamic scenarios. We introduce the RL-Enhanced Interaction Learning to enhance dynamic obstacle avoidance capability through two dual learning processes.

The perception and decision-making models explore the reasonable actions by interacting with the environment, while the reasoning model deduces the latent state distribution based on the action sequence. These two dual processes generate reward feedback for each other and collectively enhance their performance. The whole process is similar to a blind person carefully exploring the outdoor scene and always imagining the surrounding according to the movements.

As shown in Figure 3, the trained parameters from the first phase are used to initialize three models. A value network with random initialization and decision-making model form an Actor-Critic framework for proximal policy optimization(PPO)[Schulman *et al.*, 2017] update.

First, the current o_t and historical observation sequences $o_{t-n:t-1}$ are concatenated together and fed into the perception model to calculate the mean μ_t^P of the latent state distribution. Since policy module in Actor-Critic involves a sampling process, we directly treat μ_t^P as the latent state s_t without sampling. Then decision-making model(policy module) predicts the action a_t^{pred} based on s_t. After that, the environment executes the action a_t^{pred} and returns *navigation-reward* R_t^{nav} . Simultaneously, the reasoning model infers the most likely distribution of latent state N_t^R based on the current action a_t^{pred} and the historical real actions $a_{t-n:t-1}^{real}$. The probability of μ_t^P in N_t^R is treated as a *similarity-reward* R_t^{sim} for evaluating the quality of the a_t. This process can be formulated as

$$\mu_t^P = \mathrm{MLP}^P(\mathrm{LSTM}^P([o_t; o_{t-n:t-1}])) \tag{9}$$

$$a_t^{pred} = \mathrm{MLP}^D(\mu_t^P) \tag{10}$$

$$o_{t+1}, R_t^{nav} = Env(a_t^{pred}) \tag{11}$$

$$\mu_t^R, \sigma_t^R = \mathrm{MLP}^R(\mathrm{LSTM}^R([a_t^{pred}; a_{t-n:t-1}^{real}])) \tag{12}$$

$$R_t^{sim} = \mathrm{Prob}(\mu_t^P \mid N(\mu_t^R, \sigma_t^R)) \tag{13}$$

Optimization for perception and decision-making. The PPO is employed to optimize the perception and decision-making model via maximizing the expected cumulative reward($R_t^{nav} + R_t^{sim}$). The loss functions of perception and decision-making model are calculated as: $L^{P,D} = \alpha * L^{policy} + \beta * L^{value} - \eta * H$, where H, L^{policy}, L^{value} represent the policy entropy, policy loss, and value loss in PPO update, respectively, and α, β, η are hyper-parameters.

Optimization for reasoning. In order to maintain the stability of the reasoning model, we design a Cumulative-Discount-Reward Weighed loss(DRW-Loss for short) mechanism to optimize the reasoning model asynchronously. The reasoning model is updated at a lower frequency than the PPO. Firstly, we collect the $\{\mu_t^P, a_t^{pred}, R_t^{nav}\}$ generated by perception and decision-making model during the interaction. Then, we attempt to adopt the μ_t^P and a_t^{pred} as training samples to optimize the reasoning model. However, at the beginning of training, μ_t^P and a_t^{pred} are sub-optimal or even incorrect since the policy is not good enough. Such data should be discarded instead of being used to train the reasoning model, as it may cause the model to collapse. Therefore, we calculate the DRW-Loss to stabilize the training. Specifically, DRW-Loss use the *navigation-reward* R_t^{nav} to calculate the discounted reward $R_t^{discount}$. A large discount reward means corresponding action is more valuable. So we use the $R_t^{discount}$ to weight the mean squared error corresponding to the sample $\{\mu_t^P, a_t^{pred}\}$:

$$R_t^{discount} = R_t^{nav} + \gamma^1 R_{t+1}^{nav} + ... + \gamma^{T-t} R_T^{nav} \tag{14}$$

$$L_{1,t}^R = R_t^{discount} * \|\mu_t^P - s_t^R\|_2^2 \tag{15}$$

$$L_{2,t}^R = (-1 - \log \sigma_t^R + (\mu_t^R)^2 + \sigma_t^R) \tag{16}$$

$$L_t^R = L_{1,t}^R + \lambda * L_{2,t}^R \tag{17}$$

where γ, λ are hyper-parameters. The reasoning model assesses the action generated by the perception and decision-making models through the reward R_t^{sim}. Meanwhile, the perception and decision-making models produce more pair data $\{\mu_t^P,\ a_t^P\}$ with high navigation-reward to promote the

Zero-shot scene benchmark	Metrics mean/std	S_1+Density change	S_2+Shape change	S_3+Density change	S_4+Speed change	S_5+Volume change	S_6+ Obstacle change	S_7+Edge change	S_8+View change	AVG
					One-stage method					
Move-base	SR	60%	70%	65%	10%	75%	15%	65%	30%	48.75%
DM-RCA	SR	47.50%(3.7)	69.75%(4.5)	59.25%(3.3)	48.25%(3)	78%(4.6)	62.75%(6.9)	87.75%(4.7)	73.5%(5)	65.84%
	AS	229(20)	245(20)	218(44)	299(20)	226(17)	**167**(24)	210(16)	191(15)	223
PPO	SR	27%(3.4)	52%(7.8)	41%(5.8)	26.25%(6.4)	52%(8)	44.5%(4.6)	86.75%(2.1)	76%(5)	50.69%
	AS	262(22)	304(107)	521(124)	241(39)	372(98)	281(87)	240(4)	293(27)	314
					Two-stage method					
MOE-VUCA	SR	25%(2)	64%(8)	40%(3.9)	39.5%(4)	58%(9.2)	63.25%(3.3)	86%(3.6)	34%(6.1)	51.22%
	AS	**176**(28)	**194**(23)	**134**(37)	**143**(24)	**161**(46)	184(64)	**186**(73)	**98**(20)	**159**
DM-RCA*	SR	75%(1)	81.5%(3.2)	48%(2.8)	52%(7.8)	85%(1.7)	82.5%(1.3)	93.5%(0.8)	65%(7.7)	72.81%
	AS	250(15)	214(26)	276(96)	263(12)	237(9)	233(66)	204(19)	230(27)	238
PPO*	SR	29.25%(3.9)	64%(4)	35.25%(5.4)	42.75%(6.3)	50.75%(5)	49%(0.9)	93.75%(2.1)	**79.75**%(4.5)	55.56%
	AS	653(100)	376(80)	299(50)	444(67)	485(88)	198(22)	227(4)	226(15)	363
Ours	SR	**84.5%**(1.2)	**83.75**%(1.7)	**69.5%**(1.1)	**56%**(1.9)	**88**%(4.0)	**83**%(3.2)	**95.5**%(0.8)	74.5%(4.3)	**79.34**%
	AS	426(33)	386(37)	344(22)	383(56)	391(50)	239(44)	250(24)	313(18)	341

Table 1: Model comparison on *Success rate*(SR) and *Arriving step*(AS) using zero-shot scene benchmark, which contains eight modified scenarios. Each scene is a modification of the corresponding scene in few-shot-scene benchmark. The comparison algorithms (DM-RCA and PPO) are One-stage algorithms and MOE-VUCA is a Two-stage training method. For fair comparison, we also use our collected data to pre-train these One-stage methods (denoted by *). The numbers in parentheses are Standard deviation.

evolution of the reasoning model. Two dual learning processes are mutually reinforced by corresponding reward feedback, collectively improving performance.

4 Experiment

4.1 Experiment Setup

Scene benchmark. We adopt four simple scenarios to train the algorithm, including open scene, sparse scene, dense scene, and dynamic scene. To investigate obstacle avoidance and path-finding capabilities, we design two testing benchmarks. As shown in Figure 4, the first one is few-shot scene, which includes many scenarios similar to training phase. The second is the zero-shot scene, where the scenarios are quite different. We add noise to the observations and adjust the distribution of the goal-position, obstacle-shape, obstacle-speed, and obstacle-density to design zero-shot scene. Zero-shot scene benchmark is used to evaluate the robustness and generalization in new scenarios. We adopt a lightweight robot simulator ROS-Stage[1] to implement evaluation, which is more compatible with real robots than OpenAI Gym[2].

Navigation setting. The goal point is randomly selected before navigation. The observation is defined as $o_t = [o_t^L; o_t^P; o_t^V]$, which means the concatenation of the Lidar measurement o_t^L, target relative position o_t^P and robot velocity o_t^V. o_t^P is relative vectors between the robot's real-time position and the target. The action is defined as $a_t = [v_t; w_t]$, representing the Forward and Angular velocity respectively. The observation, latent state and action are all continuous high-dimensional vector. Our reward function has two components, including navigation-reward (R^{nav}) and similarity-reward (R^{sim}). The R^{nav} is feedback from the simulator, and R^{sim} is calculated by the reasoning model. We design the reward function as follows:

$$R_t = R_t^{sim}/\rho + R_t^{nav} \tag{18}$$

$$R^{\text{step}} = \|P_t^a - P_t^g\|_2 - \|P_{t-1}^a - P_{t-1}^g\|_2 \tag{19}$$

$$R_t^{\text{nav}} = \begin{cases} R^{\text{goal}} = 30, & \text{if reach} \\ R^{\text{collision}} = -20, & \text{if crash} \\ R^{\text{time}} = -0.01, & \text{if alive} \\ R^{\text{step}}, & \text{if alive} \end{cases} \tag{20}$$

where P^a and P^g mean the position of the agent and goal, and ρ is a normalized hyperparameter. We set the hyperparameter ρ equal to the probability of a zero vector in a 90-dimensional standard normal distribution.

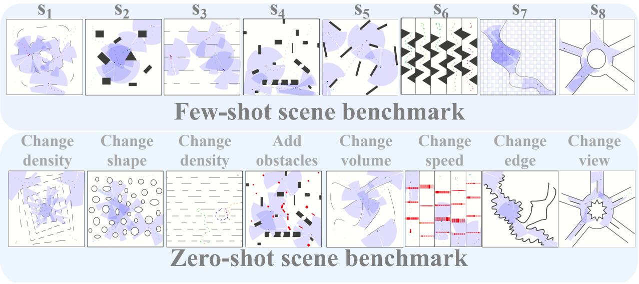


Figure 4: Some scenes in two test benchmarks. Each test scenario in the zero-shot scene benchmark is derived from the few-shot scene benchmark, with specific changes.

[1] http://wiki.ros.org/stage

[2] http://gym.openai.com/

Dateset collection. Since there is no publicly available navigation dataset, we adopt A^* algorithm to plan a global path in a static scene for the first stage of pre-training. Actually, many reasonable strategies can be used to generate the dataset. We have tried alternative methods such as manual annotation, RRT* and finally chose A* as the optimized method for dataset generation. Then we use the Timed Elastic Band (TEB) algorithm to transform trajectory generated by A* into action sequences. The TEB considers the kinematics constraints and optimizes the local action velocity based on the global trajectory. Then, we record observation-action data and manually clean up some exception data. Finally, we collect 200 trajectories, each with about 300 pair data $\{o_t, a_t\}$.

Baselines. We choose three types of algorithms for comparison. (1) Widely used robot open-source library **Move-base**[3]. (2) Learning-based navigation algorithms **DM-RCA**[Long *et al.*, 2018] and **MOE-VUCA**[Zhang *et al.*, 2021]. (3) DRL baseline algorithms **PPO**[Schulman *et al.*, 2017] with the same configuration as ours. All comparison algorithms, except the Move-base algorithm are trained in the same scene and each benchmark is evaluated 400 times.

Metrics. The optimization target of the navigation is to avoid collision as much as possible and successfully reach the goal point o_t^P in a short time. To achieve this, we adopt two metrics (*Success rate(%)*, *Arriving step*) to evaluate the performance as previous learning-based works (DM-RCA,MOE-VUCA) did. *Success rate* is a comprehensive indicator of the capability to path-finding and collision avoidance and *Arriving step* assesses the navigation efficiency.

Implementation details. We adopt Microsoft's NNI for tuning parameters in the first stage. In the second stage, we have tested several sets of parameters in a small range and chose the best parameters, and the comparison algorithms also chose the best parameters in the same scene. We set the latent state to 90 dimensions, and the sequence length n is 20, and α, β, η, λ, γ are 1.0, 20.0, 5e-4, 0.01, 0.99. We discover that updating the reasoning model once every 10 PPO updates is more appropriate. Too frequent updates lead to unstable training, yet too slow updates bring performance degradation as the reasoning model evaluates the action and generates the reward. All models are implemented using PyTorch. We use optimizer Adam with a learning rate of 1e-3 in VAE-Enhanced Demonstration learning and 3e-5 in RL-Enhanced Interaction learning. Video of real-robot experiments are shown at https://youtu.be/jD_7sCdMMWk. Datasets and part of the code will be released at https://github.com/zwq2018/CL_PDR_NAV.

4.2 Results and Analysis

As Figure 5, compared to other methods in few-shot scenes, our algorithm achieves comparable performance as others in most scenarios, except for Move-base. However, Move-base requires an environmental map in advance.

As shown in Table 1, the performance of all the methods in zero-shot scene is significantly degraded. Obviously, environmental distribution shift does have a dramatic impact, and our method is more robust and generalizable than others. In highly dynamic scenes like S_6-Obstacle change(SC_6 for short), Move-base is almost impossible to avoid the obstacle (10% in SC_4 and 15% in SC_6), but our algorithm achieves the best results(56% and 83%). In complicated maze scenes, our algorithm lead over the runner-up (DM-RCA) by a large margin (9.6% in SC_1 and 2.3% in SC_2). Additionally, SC_8 simulates a complex traffic roundabout, and our algorithm is only inferior to PPO-Pretrained method by a small margin(74.5% compared to 79.75%).

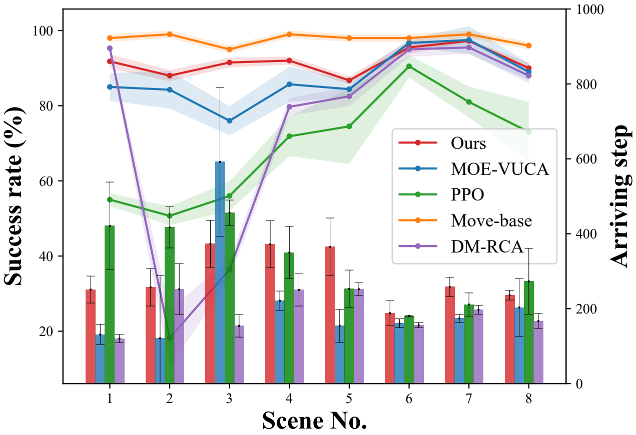


Figure 5: Evaluation on few-shot scene benchmark. It includes eight similar scenarios, the line graph represents the *Success rate*(left), and the bar chart means the *Arriving step*(right).

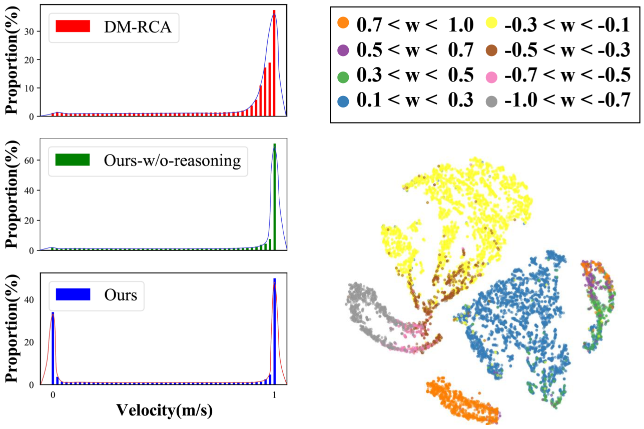
MOE-VUCA performs the worst among all methods in zero-shot scenes, although it has the shortest *Arriving step*. We guess it suffers from poor generalization due to the parameter fusion trick it used. DM-RCA is a pretty simple but effective model that greatly outperforms other sophisticated models except ours. It reveals that more sophisticated networks may suffer from the challenge of poor generalization. In addition, we investigate the effect of pre-training using our collected data. For the comparison algorithms, the performance after pre-training with our collected data (denoted by *) does not show a significant improvement over training from scratch. We also observe that the *Arriving step* of ours is much longer than other algorithms. We suspect that the reasoning model encourages the agent to generate more conservative actions for higher similarity-reward R_t^{sim}, resulting in a longer navigation step. A detailed analysis is provided in Section 4.4.

4.3 Ablation Study

We analyze the contribution of different components in our algorithm. We design three variants of the algorithm: (1)We remove the reasoning model and then train the algorithm using two-stage phases. (2)We replace the DRW-Loss with the mean square error when updating the reasoning model. (3)Without VAE-Enhanced Demonstration Learning phase. Table 2 illustrates that the reasoning model is essential to our algorithm. The introduction of DRW-Loss and VAE-Enhanced Demonstration Learning can facilitate the learning

[3] http://wiki.ros.org/move_base

of the reasoning model.



(a) (b)

Figure 6: (a) The distribution of linear velocity collected by our method shows a bimodal shape. (b) Using t-SEN to visualize the latent states by reasoning model according to different actions. Each point in the figure is a latent state vector after dimension reduction. w means Angular Velocity(rad/s) of the agent.

Model variant	Success rate
Ours	79.3%
w/o-reasoning model	55.5%
w/o-VAE-Enhanced demonstration learning	63%
w/o-DRW-Loss	55.1%

Table 2: Ablation study on different components(Average).

4.4 Analysis of the Reasoning Model

We investigate the impact of the reasoning model on actions. The outputs of the decision-making model include Forward Velocity v and Angular Velocity w. Firstly, we collect the w and v of several methods in the same scenario and analyze the corresponding behavior patterns. In Figure 6(a), the action(v) distribution output by DM-RCA are clustered at a larger velocity value($1m/s$), forming a uni-modal distribution. In contrast, our actions form a bi-modal distribution($0m/s$ and $1m/s$). Thus our strategy is more conservative, preferring to brake rather than bypass when encountering an obstacle. Obviously, comparison policies are more dangerous than ours. It shows that the reasoning model makes the system safer by constraining the action output, albeit sacrificing navigation efficiency. Secondly, we analyze the relationship between the high-dimensional latent vectors s^R and the action a^w in Figure 6(b). It visualizes the distribution of latent states after dimension reduction. The 90-dim continuous latent states s^R are calculated by reasoning model from continuous action sequences $a_{t-n:t}$. These latent vectors are clustered in different regions depending on the action before. It reveals that reasoning model maps similar actions to neighboring regions in latent space, and different actions are projected to distinct. It exhibits a clustering-like effect.

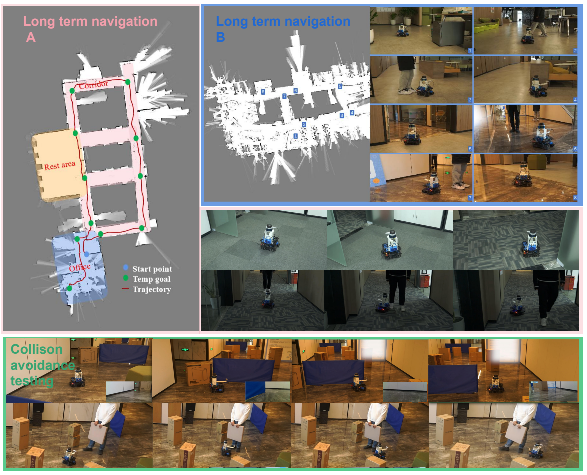


Figure 7: Real-world experiments using Turtlebot3 with ROS.

4.5 Real-World Experiment

We deploy the algorithm on a wheel robot. As shown in Figure 7, the robot navigates in a crowded and complex building with moving staff. We randomly select goals and the robot navigates to the targets one by one. The robot is equipped with a 16-line 3D-LIDAR (Velodyne-16), a depth camera (Real-Sense D435) and an edge computing device (NVIDIA Jetson AGX Xavier with Ubuntu and ROS Melodic). All computational processes are performed onboard. Despite the considerable discrepancy between the training scenes and the real scenes, the robot successfully achieve navigation without human help. It means our closed-loop reasoning mechanism can successfully transfer from a simple simulator to a real robot, due to the good adaptability of our approach.

5 Conclusion

In this work, we introduce a reasoning process to create an inverse mapping from the output action to the latent state, forming a closed loop with the perception and decision-making process. The reasoning model closes the learning processes of the perception and decision-making and implicitly facilitates the whole system to develop safer and more reasonable collision avoidance behaviors. Experiments have shown that our algorithm is more generalizable and robust in the novel scenario, surpassing the previous approaches in terms of safety and task accomplishment. Real-world robot navigation testing also confirms that its capability being deployed to the real scenes with minimal cost.

Acknowledgments

This work is supported by the Key Research and Development Program of Zhejiang Province, China (No. 2021C01013), the National Key Research and Development Project of China (No. 2018AAA0101900), CKCEST, and MOE Engineering Research Center of Digital Library.

References

[Artetxe *et al.*, 2017] Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*, 2017.

[Cao *et al.*, 2020] Ruisheng Cao, Su Zhu, Chenyu Yang, Chen Liu, Rao Ma, Yanbin Zhao, Lu Chen, and Kai Yu. Unsupervised dual paraphrasing for two-stage semantic parsing. In *Proc. of ACL*, pages 6806–6817, July 2020.

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

[Faust *et al.*, 2018] Aleksandra Faust, Kenneth Oslund, Oscar Ramirez, Anthony Francis, Lydia Tapia, Marek Fiser, and James Davidson. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5113–5120. IEEE, 2018.

[Fox *et al.*, 1997] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[Fu *et al.*, 2021] Huiqiao Fu, Kaiqiang Tang, Peng Li, Wenqi Zhang, Xinpeng Wang, Guizhou Deng, Tao Wang, and Chunlin Chen. Deep reinforcement learning for multi-contact motion planning of hexapod robots. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI), Montréal, QC, Canada*, pages 2381–2388, 2021.

[Ha and Schmidhuber, 2018] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Proc. of NeurlPS*, 2018.

[He *et al.*, 2016a] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *Proc. of NeurlPS*, 2016.

[He *et al.*, 2016b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[Ho and Ermon, 2016] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Proc. of NeurlPS*, 2016.

[Khatib, 1986] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[Kingma and Welling, 2014] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, Banff, Canada, April 2014.

[Koenig and Likhachev, 2002] Sven Koenig and Maxim Likhachev. D* lite. *Aaai/iaai*, 15:476–483, 2002.

[Long *et al.*, 2018] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6252–6259. IEEE, 2018.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Pfeiffer *et al.*, 2018] Mark Pfeiffer, Samarth Shukla, Matteo Turchetta, Cesar Cadena, Andreas Krause, Roland Siegwart, and Juan Nieto. Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. *IEEE Robotics and Automation Letters*, 3(4):4423–4430, 2018.

[Qi *et al.*, 2021] Jie Qi, Hui Yang, and Haixin Sun. Mod-rrt*: A sampling-based algorithm for robot path planning in dynamic environment. *IEEE Transactions on Industrial Electronics*, 68(8):7244–7251, 2021.

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

[Stentz and others, 1995] Anthony Stentz et al. The focussed d* algorithm for real-time replanning. In *IJCAI*, volume 95, pages 1652–1659, 1995.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Tsounis *et al.*, 2020] Vassilios Tsounis, Mitja Alge, Joonho Lee, Farbod Farshidian, and Marco Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3699–3706, 2020.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[Yang *et al.*, 2020] Chuanyu Yang, Kai Yuan, Qiuguo Zhu, Wanming Yu, and Zhibin Li. Multi-expert learning of adaptive legged locomotion. *Science Robotics*, 2020.

[Zhang *et al.*, 2021] Wenqi Zhang, Kai Zhao, Peng Li, et al. Learning to navigate in a vuca environment: Hierarchical multi-expert approach. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9254–9261, 2021.