Diffusion Policies for Short-Horizon Planning in Robot Crowd Navigation

Wendong Li

Born 3rd June 2001 in Nanchang, China

27th July 2026

Master's Thesis Mathematics

Advisor: Prof. Dr. Jochen Garcke

Second Examiner: Dr. Bastian Bohn

Institut für Numerische Simulation

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Acknowledgments

I would like to express my sincere gratitude to the people who supported me during the preparation of this thesis.

First, I would like to thank Prof. Dr. Jochen Garcke for supervising this thesis, for his guidance throughout the project, and for his time and feedback during our discussions. His broad experience and deep knowledge provided valuable direction for this work, and his continuous support helped me develop the ideas and presentation of this thesis.

I would also like to thank the friends I met at the University of Bonn during my master's studies for their help and care.

I am grateful to my girlfriend for her companionship, encouragement, patience and support throughout this period.

Finally, I would like to thank my parents for their unwavering support and love. They have always stood by me, and their trust and care have been a constant source of strength.

Contents

Acknowledgments **1**

1 Introduction **1**

2 Reinforcement Learning Background **4**
2.1 Markov Decision Processes 4
2.2 Value Functions and Bellman Equations 6
2.3 Parameterized Policies and Policy Gradients 8
2.4 Actor–Critic Methods and Advantage Estimation 12
2.5 Proximal Policy Optimization 14

3 Diffusion Models and Diffusion Policies **16**
3.1 Denoising Diffusion Models 16
3.2 Conditional Diffusion Models as Stochastic Policies 21
3.3 Expressiveness Beyond Gaussian Policies 22

4 Robot Crowd Navigation Background **26**
4.1 Crowd Navigation and Common Notation 26
4.2 Rule-Based Collision Avoidance: ORCA 28
4.3 Encoder-Based Reinforcement Learning for Crowd Navigation 29
4.4 Interaction Encoders 30
4.5 CrowdNav++ as a Strong Learning-Based Baseline 32
4.6 From State Representation to Action Generation 34

5 PDPO for Crowd Navigation **36**
5.1 Environment MDP and Diffusion Policy Specialization 36
5.2 Offline Denoising Pretraining 38
5.3 The Augmented Denoising MDP 39
5.4 Online PPO Fine-Tuning 43

6 Experiments **47**
6.1 Experimental Setup 47
6.1.1 Simulation Environment 47
6.1.2 Original and Bounded Evaluation Protocols 48
6.1.3 Baselines and Ablations 49
6.1.4 Training and Evaluation Protocol 49
6.2 Development of the Training Pipeline 50
6.3 Quantitative Evaluation 52
6.3.1 Original CrowdNav++ Benchmark 52
6.3.2 Boundary Diagnosis 52

6.3.3 Bounded CrowdNav 53
6.4 Ablation Study: Role of Action Chunks 54
6.5 Qualitative Analysis of Action Chunks 54
6.6 Scope of the Experimental Evidence 57

7 Conclusion **58**

A Implementation Details **60**
A.1 Observation Encoder 60
A.2 Diffusion Policy and Value Function 61
A.3 Training Hyperparameters 62
A.4 Model Size and Checkpoint Selection 63
A.5 Additional Training Curves 63

Chapter 1

Introduction

Background and Motivation Robot crowd navigation is a fundamental problem in robotics, where a robot must move to a target goal while avoiding collisions with pedestrians and maintaining efficient progress in a dynamic environment. Unlike navigation in static maps, the surrounding agents are also moving, and their future positions are only partially predictable. The robot therefore has to make sequential decisions under uncertainty, where a locally reasonable action may become unsafe or inefficient after only a few timesteps.

A central difficulty of crowd navigation is the multi-modal nature of local interactions. When a robot approaches a group of pedestrians, several maneuvers may be simultaneously feasible: it may pass on the left, pass on the right, slow down and yield, or move through a temporary gap. These alternatives are not merely small perturbations of the same action. They correspond to qualitatively different local navigation strategies.

This creates a representation problem for learned policies. Many continuous control methods use deterministic actions or unimodal Gaussian action distributions. Such policies are convenient for optimization, but they are not well matched to observations that admit several separated feasible decisions. A Gaussian policy may increase its variance, but it still represents the action distribution by a single mode. If two feasible strategies are separated in action space, the Gaussian mean may lie between them and need not correspond to a meaningful navigation decision.

This thesis studies diffusion policies as a more expressive policy class for robot crowd navigation. The primary motivation is the ability of diffusion models to represent non-Gaussian and multi-modal conditional action distributions. We further generate short-horizon action chunks, so that different modes of the policy can correspond to coherent local motion plans rather than isolated velocity commands. The central question is whether diffusion policies can improve dense crowd navigation by representing multi-modal local navigation strategies, and whether action chunks can strengthen this advantage by encoding these strategies as short-horizon plans.

Development of Robot Crowd Navigation Methods Classical crowd-navigation methods rely on hand-designed local interaction rules. The Social Force model represents pedestrian motion through attraction and repulsion forces, while ORCA constructs reciprocal velocity constraints for multi-agent collision avoidance [HM95; Ber+11]. These methods are efficient and interpretable, but their local and reactive nature can lead to overly conservative behavior or freezing in dense crowds [TK10].

Learning-based methods replace hand-designed interaction rules with policies trained from data or simulation. Early deep-RL approaches learn navigation behavior among dynamic agents by optimizing long-term returns [Eve+18]. Attention-based methods such as

SARL aggregate human–robot interaction features and improve crowd-aware decision-making [Che+19]. Subsequent work further improves interaction modeling through relational graph learning [Che+20], structural recurrent models [Liu+21], and enhanced spatial-attention graphs [Shi+22].

More recent crowd-navigation policies incorporate richer temporal and predictive structure. CrowdNav++ uses an attention-based interaction graph together with short-horizon human trajectory prediction [Liu+23]. Related work studies attention-based spatio-temporal graph representations for crowd behavior modeling [ZG24]. Safety-oriented extensions also consider uncertainty handling and constrained policy optimization to improve robustness under prediction errors [Yao+25].

These developments substantially improve observation representation, interaction modeling, prediction, and safety treatment. However, they do not directly address the expressiveness of the action distribution itself. In dense crowds, the same observation may support several qualitatively different but feasible local strategies. This thesis therefore studies a complementary direction: using a conditional diffusion policy to represent a flexible multi-modal distribution over local navigation behavior.

Diffusion Policies for Multi-Modal Action Generation Diffusion models provide a flexible generative modeling framework based on iterative denoising [Ho+20]. Starting from Gaussian noise, a learned reverse process gradually transforms the noise into a structured sample. When used as a conditional generative model, this reverse denoising process can represent complex, non-Gaussian, and multi-modal conditional distributions.

This makes diffusion models attractive as policy classes for robotics. A diffusion policy treats an action object as the generated sample. Conditioned on the current observation, the policy samples this object by denoising Gaussian noise. Compared with a standard Gaussian policy, this gives a richer conditional action distribution. In robotic manipulation, diffusion policies have shown strong performance by generating multi-step action sequences rather than isolated controls [Chi+25]. Recent work further extends diffusion policies to reinforcement learning by optimizing the denoising process with policy-gradient methods [Ren+25].

In this thesis, the generated action object is a short-horizon action chunk

$$x_0^t = (x_{0,1}^t, x_{0,2}^t, \ldots, x_{0,H}^t), \qquad x_{0,j}^t \in \mathbb{R}^2, \tag{1.1}$$

where each component is a two-dimensional velocity command. During execution, only the first action is applied to the robot,

$$a_t = x_{0,1}^t, \tag{1.2}$$

and a new chunk is sampled at the next timestep. This receding-horizon structure preserves closed-loop responsiveness while allowing each sampled mode of the diffusion policy to correspond to a coherent short-horizon local plan.

Planning Diffusion Policy Optimization The main method considered in this thesis is Planning Diffusion Policy Optimization (PDPO), an offline-to-online reinforcement-learning framework for crowd navigation. The environment is modeled as a Markov decision process

$$\mathcal{M}_{\text{env}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_0, \mathcal{P}, \mathcal{R}, \gamma), \tag{1.3}$$

where the state contains the robot state and the observed or predicted states of nearby pedestrians, and the environment action is a two-dimensional velocity command. At each environment timestep, an observation encoder maps the crowd-navigation state s_t to a

latent representation z_t. A conditional diffusion policy then samples an action chunk x_0^t conditioned on z_t.

PDPO follows a two-stage training procedure. First, the diffusion policy is pretrained by behavioral cloning on demonstration trajectories. Given an expert action chunk, a noisy version is obtained by a forward diffusion process, and the denoising network is trained to predict the injected Gaussian noise. This stage initializes the policy with reasonable collision-avoidance behavior.

Second, the pretrained diffusion policy is fine-tuned online using Proximal Policy Optimization (PPO). Since the final action chunk is produced through multiple reverse denoising steps, the sampling process can be viewed as an internal decision process. Each denoising transition is treated as an augmented policy step, while the environment reward is assigned after the final denoised action chunk has been generated and the first action has been executed. This augmented-MDP viewpoint provides a tractable way to apply policy-gradient optimization to the denoising policy.

The thesis also considers a bounded version of the CrowdNav benchmark. In the original setting, a robot may leave the workspace and bypass dense interactions while still being counted as successful. The bounded setting treats boundary violations as collisions, thereby providing a stricter evaluation of crowd-aware navigation behavior.

Contributions The contributions of this thesis are as follows.

1. **A multi-modal diffusion-policy formulation for crowd navigation.** The thesis studies dense crowd navigation from the perspective of action distribution expressiveness and argues that multi-modal local interactions are difficult to represent with deterministic or unimodal Gaussian policies.

2. **Action chunks as local plan representations.** The thesis uses action chunks to lift diffusion-policy outputs from individual velocity commands to short-horizon local plans, so that different sampled modes can correspond to different coherent navigation strategies.

3. **An offline-to-online PDPO framework.** The thesis presents Planning Diffusion Policy Optimization, which combines denoising behavioral-cloning pretraining with PPO fine-tuning of the reverse denoising process.

4. **An empirical study in original and bounded CrowdNav benchmarks.** The thesis evaluates PDPO in dense crowd-navigation scenarios, studies the role of action chunks, and discusses how boundary constraints affect benchmark interpretation.

Thesis Organization The remainder of the thesis is organized as follows. Chapter 2 introduces the reinforcement-learning background, including Markov decision processes, policy gradients, actor–critic methods, and PPO. Chapter 3 reviews denoising diffusion models and explains how conditional diffusion models can be used as stochastic policies with multi-modal action distributions. Chapter 4 reviews robot crowd-navigation background, focusing on ORCA as a representative rule-based baseline and CrowdNav++ as a strong learning-based baseline. Chapter 5 presents the PDPO method for robot crowd navigation, including the observation encoder, diffusion action-chunk policy, behavioral-cloning pretraining, and online PPO fine-tuning. Chapter 6 reports the experimental setup, baselines, ablations, and results on both the original and bounded CrowdNav benchmarks. Finally, Chapter 7 summarizes the findings and discusses limitations and future directions.

Chapter 2

Reinforcement Learning Background

This chapter reviews the reinforcement-learning background used in the rest of the thesis. The goal is to introduce the mathematical objects needed to formulate robot crowd navigation and to understand the policy-optimization component of Planning Diffusion Policy Optimization (PDPO). We first introduce Markov decision processes, value functions, and Bellman equations. We then discuss policy-gradient methods, actor–critic estimation, and Proximal Policy Optimization (PPO). Finally, we explain how these concepts apply to continuous-control crowd navigation. The presentation in this chapter follows standard treatments of reinforcement learning, especially Sutton and Barto [SB18], with additional references given at the beginning of each section.

2.1 Markov Decision Processes

We follow Puterman [Put94] and Sutton and Barto [SB18] in introducing the Markov decision process formalism used throughout this chapter.

Definition 2.1 (Discounted Markov decision process)**.** An infinite-horizon discounted Markov decision process is a tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_0, \mathcal{P}, \mathcal{R}, \gamma), \tag{2.1}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}_0$ is the initial-state distribution, $\mathcal{P}(\cdot \mid s, a)$ is the transition kernel, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor.

At time t, the agent observes $s_t \in \mathcal{S}$, chooses an action $a_t \in \mathcal{A}$, receives reward

$$r_t = \mathcal{R}(s_t, a_t), \tag{2.2}$$

and the next state is sampled according to

$$s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t). \tag{2.3}$$

Definition 2.2 (Markov property)**.** The transition kernel $\mathcal{P}$ satisfies the Markov property if, for every measurable set $B \subseteq \mathcal{S}$,

$$\mathbb{P}\left(s_{t+1} \in B \mid s_0, a_0, \ldots, s_t, a_t\right) = \mathcal{P}(B \mid s_t, a_t). \tag{2.4}$$

Thus, conditional on the current state and action, the next-state distribution does not depend on the previous history.

Definition 2.3 (Stochastic policy)**.** A stochastic policy is a conditional distribution

$$\pi(\cdot \mid s) \tag{2.5}$$

on $\mathcal{A}$ for each state $s \in \mathcal{S}$. We write $\pi(a \mid s)$ for its probability mass function or density. In continuous-control problems, $\mathcal{A} \subseteq \mathbb{R}^d$, and $\pi(\cdot \mid s)$ is usually represented by a density with respect to Lebesgue measure.

Once a policy π is fixed, the MDP and the policy together induce a probability law over trajectories. For a finite trajectory segment

$$\tau_{0:T} = (s_0, a_0, s_1, a_1, \ldots, s_T), \tag{2.6}$$

the corresponding trajectory density is

$$p_\pi(\tau_{0:T}) = \mathcal{P}_0(s_0) \prod_{t=0}^{T-1} \pi(a_t \mid s_t)\mathcal{P}(s_{t+1} \mid s_t, a_t). \tag{2.7}$$

If the policy is parameterized as π_θ, we write

$$p_\theta := p_{\pi_\theta} \tag{2.8}$$

for the induced trajectory law.

Definition 2.4 (Discounted return and control objective)**.** For a trajectory generated under π, the discounted return from time t is

$$G_t = \sum_{\ell=0}^{\infty} \gamma^\ell r_{t+\ell}. \tag{2.9}$$

The reinforcement-learning objective is to maximize

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi}[G_0] = \mathbb{E}_{\tau \sim p_\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]. \tag{2.10}$$

For a parameterized policy π_θ, we write

$$J(\theta) := J(\pi_\theta). \tag{2.11}$$

Remark 2.5 (Episodic tasks and absorbing terminal states). In episodic tasks, termination usually occurs at a random time depending on the trajectory, for example when the robot reaches the goal, collides, or exceeds a maximum episode length. Formally, such a termination time can be viewed as a stopping time with respect to the natural filtration of the trajectory. In this thesis, we use the standard convention of adding an absorbing terminal state s_{abs} with zero reward after termination. Under this convention, an episodic task can be written as an infinite-horizon discounted MDP, and the return remains

$$G_t = \sum_{\ell=0}^{\infty} \gamma^\ell r_{t+\ell}, \tag{2.12}$$

where all rewards after termination are zero.

2.2 Value Functions and Bellman Equations

We follow the dynamic-programming perspective of Bellman [Bel57] and the reinforcement-learning presentation of Sutton and Barto [SB18] in introducing value functions and Bellman equations. These objects summarize future rewards under a fixed policy and provide the quantities that appear in the policy-gradient theorem. They are also central in value-based reinforcement learning, where the policy is improved indirectly through estimates of value functions.

Definition 2.6 (Value functions)**.** For a fixed policy π, the state-value function is

$$V^\pi(s) = \mathbb{E}_\pi \left[G_t \mid s_t = s \right], \tag{2.13}$$

and the action-value function is

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[G_t \mid s_t = s,\ a_t = a \right]. \tag{2.14}$$

The advantage function is defined by

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s). \tag{2.15}$$

Remark 2.7 (Interpretation of the advantage). The quantity $Q^\pi(s,a)$ measures the expected return after taking action a in state s and then following policy π. The value $V^\pi(s)$ is the policy average at the same state. Hence $A^\pi(s,a)$ measures whether action a is better or worse than the typical action sampled from $\pi(\cdot \mid s)$. By construction,

$$\mathbb{E}_{a\sim\pi(\cdot|s)} \left[A^\pi(s,a) \right] = 0. \tag{2.16}$$

Definition 2.8 (Bellman expectation operators)**.** For a fixed policy π, define the Bellman expectation operator for state-value functions by

$$(\mathcal{T}^\pi V)(s) = \mathbb{E}_{a\sim\pi(\cdot|s)} \left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)} V(s') \right], \tag{2.17}$$

and the Bellman expectation operator for action-value functions by

$$(\mathcal{T}^\pi Q)(s,a) = \mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)} \mathbb{E}_{a'\sim\pi(\cdot|s')} Q(s',a'). \tag{2.18}$$

Proposition 2.9 (Bellman expectation equations)**.** *For any fixed policy π, the value functions satisfy*

$$V^\pi = \mathcal{T}^\pi V^\pi, \qquad Q^\pi = \mathcal{T}^\pi Q^\pi. \tag{2.19}$$

Equivalently,

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)} \left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)} V^\pi(s') \right], \tag{2.20}$$

$$Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)} \mathbb{E}_{a'\sim\pi(\cdot|s')} Q^\pi(s',a'). \tag{2.21}$$

Proof. Using the decomposition

$$G_t = r_t + \gamma G_{t+1}, \tag{2.22}$$

we obtain

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi \left[r_t + \gamma G_{t+1} \mid s_t = s \right] \\ &= \mathbb{E}_{a\sim\pi(\cdot|s)} \left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)} V^\pi(s') \right]. \end{aligned} \tag{2.23}$$

This proves (2.20). Similarly, conditioning additionally on $a_t = a$ gives

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi \left[r_t + \gamma G_{t+1} \mid s_t = s,\ a_t = a \right] \\ &= \mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)} \mathbb{E}_{a'\sim\pi(\cdot|s')} Q^\pi(s',a'), \end{aligned} \tag{2.24}$$

which proves (2.21). □

Definition 2.10 (Optimal value functions)**.** The optimal state-value and action-value functions are defined by

$$V^*(s) = \sup_\pi V^\pi(s), \qquad Q^*(s,a) = \sup_\pi Q^\pi(s,a). \tag{2.25}$$

Definition 2.11 (Bellman optimality operator)**.** The Bellman optimality operator for action-value functions is defined by

$$(\mathcal{T}^* Q)(s,a) = \mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)} \left[\sup_{a' \in \mathcal{A}} Q(s',a') \right]. \tag{2.26}$$

Proposition 2.12 (Bellman optimality equation)**.** *The optimal action-value function satisfies*

$$Q^* = \mathcal{T}^* Q^*, \tag{2.27}$$

that is,

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)} \left[\sup_{a' \in \mathcal{A}} Q^*(s',a') \right]. \tag{2.28}$$

Remark 2.13 (Policy evaluation and control). The Bellman expectation equation is associated with policy evaluation: for a fixed policy π, one tries to estimate V^π or Q^π. The Bellman optimality equation is associated with control: one tries to estimate Q^* and then chooses actions greedily with respect to this estimate. Thus value-based methods learn policies indirectly through value functions.

Temporal-difference learning was introduced in [Sut88], and Q-learning is the classical off-policy temporal-difference control method introduced in [WD92].

Definition 2.14 (Temporal-difference error)**.** Let F_ω be an approximation of a value function, where $F_\omega(s) = V_\omega(s)$ for state-value learning and $F_\omega(s,a) = Q_\omega(s,a)$ for action-value learning. Given a one-step bootstrapped target y_t, the temporal-difference error is

$$\delta_t = y_t - F_\omega(x_t), \tag{2.29}$$

where $x_t = s_t$ or $x_t = (s_t, a_t)$, respectively. Common choices of the target are

$$y_t^V = r_t + \gamma V_\omega(s_{t+1}), \tag{2.30}$$

$$y_t^{\mathrm{SARSA}} = r_t + \gamma Q_\omega(s_{t+1}, a_{t+1}), \qquad a_{t+1} \sim \pi(\cdot \mid s_{t+1}), \tag{2.31}$$

$$y_t^{\mathrm{Q}} = r_t + \gamma \sup_{a' \in \mathcal{A}} Q_\omega(s_{t+1}, a'). \tag{2.32}$$

The first target is used for state-value policy evaluation, the second for on-policy action-value learning, and the third for Q-learning.

Remark 2.15 (Temporal-difference error as Bellman residual). The temporal-difference error is a sample-level Bellman residual. For example, (2.30) compares the current estimate $V_\omega(s_t)$ with the bootstrapped target $r_t + \gamma V_\omega(s_{t+1})$. Thus temporal-difference learning can be viewed as stochastic approximation to a Bellman fixed-point equation.

Definition 2.16 (Value-based learning paradigm)**.** A value-based reinforcement-learning method learns an approximation to V^π, Q^π, or Q^* using temporal-difference targets, and then derives a policy from this value estimate. In particular, Q-learning aims to approximate Q^* and selects actions greedily with respect to the learned action-value function.

Algorithm 1 Generic temporal-difference value learning

Require: Value approximation F_ω, behavior policy μ, learning rate α

Initialize parameters ω
for each episode **do**
  Sample $s_0 \sim \mathcal{P}_0$
  for $t = 0, 1, 2, \dots$ **do**
    Sample $a_t \sim \mu(\cdot \mid s_t)$ and observe (r_t, s_{t+1})
    Choose a Bellman target y_t
    Set $x_t = s_t$ for value learning or $x_t = (s_t, a_t)$ for action-value learning
    Compute $\delta_t = y_t - F_\omega(x_t)$
    Update

$$\omega \leftarrow \omega + \alpha \delta_t \nabla_\omega F_\omega(x_t)$$

    if s_{t+1} is terminal **then**
      break
    end if
  end for
end for

For instance, the tabular Q-learning update is

$$Q_{k+1}(s_t, a_t) = Q_k(s_t, a_t) + \alpha_k \left[r_t + \gamma \sup_{a' \in \mathcal{A}} Q_k(s_{t+1}, a') - Q_k(s_t, a_t) \right], \tag{2.33}$$

where $\alpha_k > 0$ is a learning rate. With function approximation, the same principle is implemented by updating the parameters ω to reduce the squared temporal-difference error.

Remark 2.17 (Connection to policy-gradient methods). Value-based methods learn a value function as the main object and derive a policy from it, for example by greedy action selection. Policy-gradient methods instead optimize π_θ directly. Nevertheless, value functions still play a central auxiliary role: they provide estimates of Q^{π_θ} or A^{π_θ} in the policy-gradient theorem. In actor–critic methods, the critic is such a learned value function, and its temporal-difference error is often used as a sample-level advantage estimate.

2.3 Parameterized Policies and Policy Gradients

We follow the likelihood-ratio formulation of Williams [Wil92] and the policy-gradient theorem of Sutton et al. [Sut+99] in presenting policy-gradient methods. Policy-gradient methods optimize a parameterized policy directly. In this section we derive the policy-gradient theorem, which is the mathematical basis for the actor–critic and PPO methods introduced in the following sections. The central idea is that the policy parameters affect the expected return through the trajectory distribution induced by the policy.

Definition 2.18 (Parameterized stochastic policy)**.** Let $\Theta \subseteq \mathbb{R}^m$ be a parameter space. A parameterized stochastic policy is a family

$$\{\pi_\theta(\cdot \mid s) : s \in \mathcal{S}, \theta \in \Theta\} \tag{2.34}$$

of stochastic policies indexed by θ. We write p_θ for the trajectory law induced by π_θ, as introduced in Section 2.1.

Assumption 2.19 (Regularity)**.** We assume that $\pi_\theta(a \mid s)$ is differentiable with respect to θ, that its support does not depend on θ, and that differentiation may be interchanged with integration in the expressions below. We also assume that rewards are bounded, so that the discounted return is integrable.

Lemma 2.20 (Score identity for trajectories)**.** *Under Assumption 2.19, for any finite horizon T,*

$$\nabla_\theta \log p_\theta(\tau_{0:T}) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t). \tag{2.35}$$

Proof. Taking logarithms in (2.7) with $\pi = \pi_\theta$ gives

$$\log p_\theta(\tau_{0:T}) = \log \mathcal{P}_0(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t \mid s_t) + \sum_{t=0}^{T-1} \log \mathcal{P}(s_{t+1} \mid s_t, a_t). \tag{2.36}$$

Since neither $\mathcal{P}_0$ nor $\mathcal{P}$ depends on θ, differentiating with respect to θ yields (2.35). □

Lemma 2.21 (Likelihood-ratio identity)**.** *Let $F(\tau_{0:T})$ be an integrable function of a finite trajectory segment. Then*

$$\nabla_\theta \mathbb{E}_{\tau_{0:T}\sim p_\theta}\left[F(\tau_{0:T})\right] = \mathbb{E}_{\tau_{0:T}\sim p_\theta}\left[F(\tau_{0:T}) \nabla_\theta \log p_\theta(\tau_{0:T})\right]. \tag{2.37}$$

Proof. Using the definition of expectation and the identity $\nabla_\theta p_\theta = p_\theta \nabla_\theta \log p_\theta$, we obtain

$$\begin{aligned} \nabla_\theta \mathbb{E}_{\tau_{0:T}\sim p_\theta}\left[F(\tau_{0:T})\right] &= \nabla_\theta \int p_\theta(\tau_{0:T}) F(\tau_{0:T}) d\tau_{0:T} \\ &= \int p_\theta(\tau_{0:T}) \nabla_\theta \log p_\theta(\tau_{0:T}) F(\tau_{0:T}) d\tau_{0:T}. \end{aligned} \tag{2.38}$$

This proves the claim. □

Lemma 2.22 (Zero-mean score)**.** *For any state s and any function $b : \mathcal{S} \to \mathbb{R}$ independent of the action,*

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a \mid s) b(s)\right] = 0. \tag{2.39}$$

Proof. For continuous action spaces,

$$\begin{aligned} \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a \mid s) b(s)\right] &= b(s) \int_{\mathcal{A}} \pi_\theta(a \mid s) \nabla_\theta \log \pi_\theta(a \mid s) da \\ &= b(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) da \\ &= b(s) \nabla_\theta \int_{\mathcal{A}} \pi_\theta(a \mid s) da = 0. \end{aligned} \tag{2.40}$$

The discrete-action case is identical, with the integral replaced by a sum. □

Lemma 2.23 (Causality)**.** *For $k < t$,*

$$\mathbb{E}_{\tau\sim p_\theta}\left[\gamma^k r_k \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] = 0. \tag{2.41}$$

Consequently, in the policy-gradient expression, the score of the action a_t may be multiplied only by rewards obtained at times $k \geq t$.

Proof. Let

$$\mathcal{F}_t = \sigma(s_0, a_0, \dots, s_t) \tag{2.42}$$

be the information available before sampling a_t. If $k < t$, then $\gamma^k r_k$ is $\mathcal{F}_t$-measurable. By the tower property of conditional expectation,

$$\begin{aligned}&\mathbb{E}\left[\gamma^k r_k \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \\ &\quad = \mathbb{E}\left[\gamma^k r_k \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \mid \mathcal{F}_t\right]\right].\end{aligned} \tag{2.43}$$

Conditioned on $\mathcal{F}_t$, the state s_t is fixed and $a_t \sim \pi_\theta(\cdot \mid s_t)$. By Lemma 2.22,

$$\mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \mid \mathcal{F}_t\right] = 0. \tag{2.44}$$

Therefore the whole expectation is zero. □

Proposition 2.24 (Finite-horizon policy-gradient identity)**.** *Define the finite-horizon objective*

$$J_T(\theta) = \mathbb{E}_{\tau_{0:T} \sim p_\theta}\left[\sum_{k=0}^{T-1} \gamma^k r_k\right]. \tag{2.45}$$

Then

$$\nabla_\theta J_T(\theta) = \mathbb{E}_{\tau_{0:T} \sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{k=t}^{T-1} \gamma^k r_k\right]. \tag{2.46}$$

Equivalently, if

$$G_t^{(T)} = \sum_{\ell=0}^{T-1-t} \gamma^\ell r_{t+\ell}, \tag{2.47}$$

then

$$\nabla_\theta J_T(\theta) = \mathbb{E}_{\tau_{0:T} \sim p_\theta}\left[\sum_{t=0}^{T-1} \gamma^t \nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t^{(T)}\right]. \tag{2.48}$$

Proof. Applying Lemma 2.21 to

$$F(\tau_{0:T}) = \sum_{k=0}^{T-1} \gamma^k r_k \tag{2.49}$$

and using Lemma 2.20, we obtain

$$\begin{aligned}\nabla_\theta J_T(\theta) &= \mathbb{E}\left[\left(\sum_{k=0}^{T-1} \gamma^k r_k\right)\left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right)\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{k=0}^{T-1} \gamma^k r_k\right].\end{aligned} \tag{2.50}$$

By Lemma 2.23, all terms with $k < t$ have zero expectation. Therefore

$$\nabla_\theta J_T(\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{k=t}^{T-1} \gamma^k r_k\right]. \tag{2.51}$$

Since

$$\sum_{k=t}^{T-1} \gamma^k r_k = \gamma^t \sum_{\ell=0}^{T-1-t} \gamma^\ell r_{t+\ell} = \gamma^t G_t^{(T)}, \tag{2.52}$$

the second form follows. □

Theorem 2.25 (Policy-gradient theorem)**.** *Under Assumption 2.19, the gradient of the discounted objective (2.11) is*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{\infty}\gamma^t \nabla_\theta \log \pi_\theta(a_t \mid s_t) Q^{\pi_\theta}(s_t, a_t)\right]. \tag{2.53}$$

Proof. By Proposition 2.24,

$$\nabla_\theta J_T(\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1}\gamma^t \nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t^{(T)}\right]. \tag{2.54}$$

Taking the limit $T \to \infty$ is justified by bounded rewards and the regularity assumptions. This gives

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t \nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t\right]. \tag{2.55}$$

Conditioning on (s_t, a_t) and using the definition of Q^{π_θ} yields

$$\begin{aligned}&\mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t \mid s_t, a_t\right] \\ &\quad = \nabla_\theta \log \pi_\theta(a_t \mid s_t) Q^{\pi_\theta}(s_t, a_t).\end{aligned} \tag{2.56}$$

Substituting this identity into the previous expression proves (2.53). □

Definition 2.26 (Discounted state-visitation distribution)**.** The discounted state-visitation distribution induced by π_θ and $\mathcal{P}_0$ is

$$d_{\mathcal{P}_0}^{\pi_\theta}(B) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbb{P}_{\pi_\theta}(s_t \in B), \qquad B \subseteq \mathcal{S}. \tag{2.57}$$

When a density exists, we write it as $d_{\mathcal{P}_0}^{\pi_\theta}(s)$.

Corollary 2.27 (State-action form of the policy-gradient theorem)**.** *With $d_{\mathcal{P}_0}^{\pi_\theta}$ defined in Definition 2.26,*

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\mathcal{P}_0}^{\pi_\theta},\ a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi_\theta}(s, a)\right]. \tag{2.58}$$

Proof. By the definition of $d_{\mathcal{P}_0}^{\pi_\theta}$,

$$\begin{aligned}&\frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\mathcal{P}_0}^{\pi_\theta},\ a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi_\theta}(s, a)\right] \\ &\quad = \sum_{t=0}^{\infty}\gamma^t \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) Q^{\pi_\theta}(s_t, a_t)\right],\end{aligned} \tag{2.59}$$

which is exactly (2.53). □

Proposition 2.28 (Baseline invariance)**.** *Let $b : \mathcal{S} \to \mathbb{R}$ be any state-dependent baseline. Then*

$$\mathbb{E}_{s\sim d_{\mathcal{P}_0}^{\pi_\theta},\ a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a \mid s) b(s)\right] = 0. \tag{2.60}$$

Consequently,

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\mathcal{P}_0}^{\pi_\theta},\ a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a \mid s)\left(Q^{\pi_\theta}(s, a) - b(s)\right)\right]. \tag{2.61}$$

Proof. For each fixed state s, Lemma 2.22 gives

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a \mid s) b(s)\right] = 0. \tag{2.62}$$

Taking the expectation over $s \sim d_{\mathcal{P}_0}^{\pi_\theta}$ proves (2.60). Subtracting this zero term from (2.58) gives (2.61). □

Baselines are commonly interpreted as variance-reduction control variates in policy-gradient estimation [Gre+04].

Corollary 2.29 (Advantage form of the policy-gradient theorem)**.** *Taking $b(s) = V^{\pi_\theta}(s)$ in Proposition 2.28, we obtain*

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\mathcal{P}_0}^{\pi_\theta},\ a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a \mid s) A^{\pi_\theta}(s,a)\right]. \tag{2.63}$$

Equation (2.63) is the form of the policy-gradient theorem used in most practical actor–critic algorithms. It separates the update direction into two factors: the score $\nabla_\theta \log \pi_\theta(a \mid s)$ of the sampled action, and the advantage $A^{\pi_\theta}(s,a)$, which measures the quality of that action relative to the current policy. In practice, the exact advantage is unknown and must be estimated from sampled trajectories. The next section explains how actor–critic methods construct such estimates using a learned critic, and Section 2.5 then introduces PPO as a stabilized actor–critic update based on a clipped empirical policy-gradient objective.

2.4 Actor–Critic Methods and Advantage Estimation

We follow Konda and Tsitsiklis [KT00] for the actor–critic framework and Schulman et al. [Sch+16] for generalized advantage estimation. The policy-gradient theorem gives a direction for improving the policy, but it depends on the unknown value quantities Q^{π_θ} or A^{π_θ}. Actor–critic methods address this by combining two function approximators. The actor is the policy $\pi_\theta(a \mid s)$, while the critic is a value function approximation $V_\phi(s)$ with parameters ϕ.

Definition 2.30 (Temporal-difference residual for the critic)**.** Given a transition (s_t, a_t, r_t, s_{t+1}), the one-step temporal-difference residual of the critic is

$$\delta_t^\phi = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t). \tag{2.64}$$

Proposition 2.31 (TD residual as an advantage estimator)**.** *If the critic is exact, i.e. $V_\phi = V^{\pi_\theta}$, then*

$$\mathbb{E}\left[\delta_t^\phi \mid s_t = s,\ a_t = a\right] = A^{\pi_\theta}(s,a). \tag{2.65}$$

Proof. If $V_\phi = V^{\pi_\theta}$, then

$$\begin{aligned}\mathbb{E}\left[\delta_t^\phi \mid s_t = s,\ a_t = a\right] &= \mathcal{R}(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}V^{\pi_\theta}(s') - V^{\pi_\theta}(s) \\ &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) = A^{\pi_\theta}(s,a).\end{aligned} \tag{2.66}$$

□

Definition 2.32 (Generalized advantage estimation)**.** Generalized advantage estimation (GAE) defines

$$\widehat{A}_t^{\mathrm{GAE}} = \sum_{\ell=0}^{T-t}(\gamma\lambda)^\ell \delta_{t+\ell}^\phi, \qquad \lambda \in [0,1], \tag{2.67}$$

for a rollout segment of length T, where δ_t^ϕ is the TD residual in (2.64). The corresponding value target is often taken to be

$$\widehat{R}_t = \widehat{A}_t^{\mathrm{GAE}} + V_\phi(s_t). \tag{2.68}$$

The parameter λ controls the bias–variance trade-off. Smaller values rely more heavily on one-step bootstrapping, while larger values use longer return information and approach a Monte Carlo estimate.

Definition 2.33 (Actor and critic losses)**.** Given rollout data generated by the current policy, the actor objective is approximated by

$$\widehat{\mathcal{J}}_{\mathrm{AC}}(\theta) = \frac{1}{M}\sum_{t=1}^{M} \log \pi_\theta(a_t \mid s_t)\widehat{A}_t, \tag{2.69}$$

where $\widehat{A}_t$ is an advantage estimate, for example $\widehat{A}_t^{\mathrm{GAE}}$. The critic is trained by minimizing

$$\mathcal{L}_{\mathrm{value}}(\phi) = \frac{1}{M}\sum_{t=1}^{M} \Big(V_\phi(s_t) - \widehat{R}_t\Big)^2. \tag{2.70}$$

Algorithm 2 Generic actor–critic with advantage estimation

Require: Initial actor parameters θ, critic parameters ϕ, learning rates $\alpha_\theta, \alpha_\phi$

for each iteration **do**

  Collect rollout data $\{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{M}$ using π_θ

  for $t = M, M-1, \ldots, 1$ **do**

    Compute TD residual

$$\delta_t^\phi = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

    Compute an advantage estimate $\widehat{A}_t$, e.g. by GAE

$$\widehat{A}_t = \sum_{\ell=0}^{M-t} (\gamma\lambda)^\ell \delta_{t+\ell}^\phi$$

    Set value target $\widehat{R}_t = \widehat{A}_t + V_\phi(s_t)$

  end for

  Update the actor by gradient ascent:

$$\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \left[\frac{1}{M}\sum_{t=1}^{M} \log \pi_\theta(a_t \mid s_t)\widehat{A}_t\right]$$

  Update the critic by gradient descent:

$$\phi \leftarrow \phi - \alpha_\phi \nabla_\phi \left[\frac{1}{M}\sum_{t=1}^{M} \Big(V_\phi(s_t) - \widehat{R}_t\Big)^2\right]$$

end for

Actor–critic methods therefore turn the theoretical gradient formula (2.63) into a practical learning procedure. The actor uses the score term $\nabla_\theta \log \pi_\theta(a_t \mid s_t)$, while the critic supplies an estimate of the advantage. PPO, introduced next, keeps this actor–critic structure but changes the actor objective to avoid overly large policy updates.

2.5 Proximal Policy Optimization

We follow Schulman et al. [Sch+17] in presenting Proximal Policy Optimization (PPO). PPO is an actor–critic policy-gradient method designed to improve stability by limiting the size of each policy update. Its starting point is the advantage form of the policy-gradient theorem, but the policy is updated through a clipped surrogate objective rather than the plain actor objective in (2.69).

Suppose rollout data are collected using an old policy $\pi_{\theta_{\text{old}}}$. For a candidate new policy π_θ, define the likelihood ratio

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}. \tag{2.71}$$

The ratio $r_t(\theta)$ measures how much the new policy changes the probability of the sampled action compared with the policy that generated the data.

Definition 2.34 (Clipped PPO objective)**.** Given advantage estimates $\widehat{A}_t$, the clipped PPO surrogate objective is

$$\mathcal{L}_{\text{clip}}(\theta) = \frac{1}{M}\sum_{t=1}^{M} \min\left\{ r_t(\theta)\widehat{A}_t, \operatorname{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\widehat{A}_t \right\}, \tag{2.72}$$

where $\epsilon > 0$ is a clipping parameter.

If $\widehat{A}_t > 0$, increasing the probability of a_t is beneficial, but the clipping prevents the ratio $r_t(\theta)$ from becoming too large. If $\widehat{A}_t < 0$, decreasing the probability of a_t is beneficial, but the clipping prevents the probability from being reduced too aggressively. Thus PPO encourages policy improvement while discouraging large policy changes.

In practical implementations, PPO combines the clipped actor objective, the critic loss, and an entropy bonus:

$$\mathcal{L}_{\text{PPO}}(\theta, \phi) = \mathcal{L}_{\text{clip}}(\theta) - c_v\mathcal{L}_{\text{value}}(\phi) + c_e\mathcal{H}(\pi_\theta), \tag{2.73}$$

where $c_v, c_e \geq 0$ are coefficients and $\mathcal{H}(\pi_\theta)$ denotes the empirical policy entropy. The value term trains the critic, while the entropy term encourages exploration.

Algorithm 3 Proximal Policy Optimization

Require: Initial actor parameters θ, critic parameters ϕ, clipping parameter ϵ, learning rates $\alpha_\theta, \alpha_\phi$

for each iteration **do**
  Set $\theta_{\text{old}} \leftarrow \theta$
  Collect rollout data $\{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{M}$ using $\pi_{\theta_{\text{old}}}$
  Compute TD residuals δ_t^{ϕ} and advantage estimates $\widehat{A}_t$
  Compute value targets $\widehat{R}_t = \widehat{A}_t + V_\phi(s_t)$
  for several optimization epochs **do**
    Compute likelihood ratios

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

    Maximize the clipped actor objective

$$\mathcal{L}_{\text{clip}}(\theta) = \frac{1}{M} \sum_{t=1}^{M} \min \left\{ r_t(\theta)\widehat{A}_t, \operatorname{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \widehat{A}_t \right\}$$

    Minimize the critic loss

$$\mathcal{L}_{\text{value}}(\phi) = \frac{1}{M} \sum_{t=1}^{M} \left(V_\phi(s_t) - \widehat{R}_t \right)^2$$

    Update θ and ϕ using stochastic gradient steps on

$$\mathcal{L}_{\text{PPO}}(\theta, \phi) = \mathcal{L}_{\text{clip}}(\theta) - c_v \mathcal{L}_{\text{value}}(\phi) + c_e \mathcal{H}(\pi_\theta)$$

  end for
end for

PPO can be viewed as a stabilized actor–critic method. It uses the critic to estimate advantages as in Section 2.4, but replaces the plain policy-gradient objective by the clipped surrogate in (2.72). This is the policy-optimization component used later in the thesis.

Chapter 3

Diffusion Models and Diffusion Policies

This chapter introduces diffusion models and explains how they become diffusion policies for continuous control. We follow Sohl-Dickstein et al. [Soh+15] for the viewpoint of learning a reverse process for a gradual noising procedure, and Ho et al. [Ho+20] for the modern discrete-time denoising diffusion probabilistic model formulation. The main viewpoint of this chapter is that a stochastic policy, as defined in Definition 2.3, is a conditional distribution over actions. Hence, choosing a policy class amounts to choosing a family of conditional action distributions. Gaussian policies provide a simple and tractable family, but they impose a unimodal geometry on the action distribution. Diffusion policies replace this restrictive density family by a conditional denoising generative model, thereby inducing a richer stochastic policy over actions or action sequences.

Throughout this chapter, the diffusion index is denoted by $k \in \{0, 1, \dots, K\}$, whereas the environment time index is denoted by t. These two indices have different meanings. The variable x_0 denotes a clean sample, and x_k denotes its noisy version after k diffusion steps. When a diffusion model is used as a policy, the clean sample x_0^t generated at environment time t will be an action object. In the crowd-navigation setting considered later, this action object is an action chunk

$$x_0^t = (x_{0,1}^t, \dots, x_{0,H}^t) \in \mathbb{R}^{Hd_a}, \qquad x_{0,j}^t \in \mathbb{R}^{d_a}, \tag{3.1}$$

where $d_a = 2$ because the robot action is a two-dimensional velocity command.

3.1 Denoising Diffusion Models

We follow the discrete-time DDPM formulation of Ho et al. [Ho+20] in presenting denoising diffusion models. Diffusion models are generative models that learn to sample from a data distribution by reversing a gradual noising process. Let p_{data} be a distribution on $\mathbb{R}^d$, and let $x_0 \sim p_{\text{data}}$ be a clean sample. The forward process progressively corrupts x_0 into noise. The reverse process is learned and transforms noise back into a structured sample.

Definition 3.1 (Forward diffusion process)**.** Let $\beta_1, \dots, \beta_K \in (0, 1)$ be a variance schedule and define

$$\alpha_k = 1 - \beta_k, \qquad \bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i, \qquad \bar{\alpha}_0 = 1. \tag{3.2}$$

The forward diffusion process is the Gaussian Markov chain

$$q(x_1, \dots, x_K \mid x_0) = \prod_{k=1}^{K} q(x_k \mid x_{k-1}), \tag{3.3}$$

with transitions

$$q(x_k \mid x_{k-1}) = \mathcal{N}(x_k; \sqrt{\alpha_k} x_{k-1}, \beta_k I), \qquad k = 1, \dots, K. \tag{3.4}$$

Equivalently,

$$x_k = \sqrt{\alpha_k} x_{k-1} + \sqrt{\beta_k} \epsilon_k, \qquad \epsilon_k \sim \mathcal{N}(0, I). \tag{3.5}$$

The forward process is fixed and contains no trainable parameters. For a suitable variance schedule, repeated application of (3.4) makes x_K close to standard Gaussian noise. Generation can then start from $x_K \sim \mathcal{N}(0, I)$ and proceed backward.

A key property of the Gaussian forward process is that the marginal distribution $q(x_k \mid x_0)$ has a closed form. This makes training efficient, because a noisy sample at any diffusion level k can be generated directly from x_0.

Lemma 3.2 (Closed form of the forward marginal)**.** *For the forward diffusion process in Definition 3.1,*

$$q(x_k \mid x_0) = \mathcal{N}(x_k; \sqrt{\bar{\alpha}_k} x_0, (1 - \bar{\alpha}_k) I). \tag{3.6}$$

Equivalently,

$$x_k = \sqrt{\bar{\alpha}_k} x_0 + \sqrt{1 - \bar{\alpha}_k} \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{3.7}$$

Proof. The claim is proved by induction. For $k = 1$, it follows directly from (3.4). Suppose it holds for $k - 1$. Then

$$x_{k-1} = \sqrt{\bar{\alpha}_{k-1}} x_0 + \sqrt{1 - \bar{\alpha}_{k-1}} \epsilon', \qquad \epsilon' \sim \mathcal{N}(0, I). \tag{3.8}$$

Using (3.5),

$$x_k = \sqrt{\alpha_k} x_{k-1} + \sqrt{\beta_k} \epsilon_k \tag{3.9}$$

$$= \sqrt{\alpha_k \bar{\alpha}_{k-1}} x_0 + \sqrt{\alpha_k (1 - \bar{\alpha}_{k-1})} \epsilon' + \sqrt{\beta_k} \epsilon_k. \tag{3.10}$$

The two Gaussian noise terms are independent, so their sum is Gaussian with covariance

$$\alpha_k (1 - \bar{\alpha}_{k-1}) I + \beta_k I = (1 - \bar{\alpha}_k) I. \tag{3.11}$$

The mean is $\sqrt{\bar{\alpha}_k} x_0$, which proves the result. □

The generative direction is the reverse of the forward noising process. Ideally, one would like to sample from the true reverse conditional distribution

$$q(x_{k-1} \mid x_k), \qquad k = K, \dots, 1. \tag{3.12}$$

This conditional depends on the unknown data distribution and is not available in closed form. A tractable intermediate object is the posterior

$$q(x_{k-1} \mid x_k, x_0), \tag{3.13}$$

where the clean sample x_0 is assumed to be known. Unlike $q(x_{k-1} \mid x_k)$, this posterior is determined entirely by the Gaussian forward process and has a closed-form expression.

Lemma 3.3 (Forward posterior)**.** *For the forward diffusion process in Definition 3.1, the posterior distribution $q(x_{k-1} \mid x_k, x_0)$ is Gaussian:*

$$q(x_{k-1} \mid x_k, x_0) = \mathcal{N}(x_{k-1}; \widetilde{\mu}_k(x_k, x_0), \widetilde{\beta}_k I), \tag{3.14}$$

where

$$\widetilde{\mu}_k(x_k, x_0) = \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k} x_0 + \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k} x_k, \tag{3.15}$$

$$\widetilde{\beta}_k = \frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k}\beta_k. \tag{3.16}$$

Proof. By Bayes' rule, as a function of x_{k-1},

$$q(x_{k-1} \mid x_k, x_0) \propto q(x_k \mid x_{k-1}) q(x_{k-1} \mid x_0). \tag{3.17}$$

Using (3.4) and Lemma 3.2, we have

$$q(x_k \mid x_{k-1}) = \mathcal{N}(x_k; \sqrt{\alpha_k} x_{k-1}, \beta_k I), \tag{3.18}$$

$$q(x_{k-1} \mid x_0) = \mathcal{N}(x_{k-1}; \sqrt{\bar{\alpha}_{k-1}} x_0, (1-\bar{\alpha}_{k-1}) I). \tag{3.19}$$

Let $y = x_{k-1}$. Ignoring constants independent of y, the negative log-density is

$$\frac{1}{2\beta_k}\|x_k - \sqrt{\alpha_k} y\|_2^2 + \frac{1}{2(1-\bar{\alpha}_{k-1})}\|y - \sqrt{\bar{\alpha}_{k-1}} x_0\|_2^2 \tag{3.20}$$

$$= \frac{1}{2} y^\top \left(\frac{\alpha_k}{\beta_k} + \frac{1}{1-\bar{\alpha}_{k-1}}\right) y - y^\top \left(\frac{\sqrt{\alpha_k}}{\beta_k} x_k + \frac{\sqrt{\bar{\alpha}_{k-1}}}{1-\bar{\alpha}_{k-1}} x_0\right) + \text{const.} \tag{3.21}$$

Therefore the posterior precision is

$$\left(\frac{\alpha_k}{\beta_k} + \frac{1}{1-\bar{\alpha}_{k-1}}\right) I = \frac{1-\bar{\alpha}_k}{\beta_k(1-\bar{\alpha}_{k-1})} I. \tag{3.22}$$

Hence the posterior variance is

$$\widetilde{\beta}_k I = \frac{\beta_k(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k} I. \tag{3.23}$$

The posterior mean is the posterior variance multiplied by the linear coefficient:

$$\widetilde{\mu}_k(x_k, x_0) = \widetilde{\beta}_k \left(\frac{\sqrt{\alpha_k}}{\beta_k} x_k + \frac{\sqrt{\bar{\alpha}_{k-1}}}{1-\bar{\alpha}_{k-1}} x_0\right) \tag{3.24}$$

$$= \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k} x_k + \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k} x_0. \tag{3.25}$$

This proves the claimed formula. □

Lemma 3.3 shows that the oracle reverse step is Gaussian when the clean sample x_0 is known. This motivates modeling each learned reverse transition as a Gaussian. However, the posterior mean $\widetilde{\mu}_k(x_k, x_0)$ still depends on x_0, which is unavailable during generation. The next result rewrites this posterior mean in terms of the noise component in x_k.

Proposition 3.4 (Noise parameterization of the reverse mean)**.** *Let x_k be generated from x_0 according to (3.7), namely*

$$x_k = \sqrt{\bar{\alpha}_k} x_0 + \sqrt{1-\bar{\alpha}_k}\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{3.26}$$

Then the posterior mean in Lemma 3.3 can be written as

$$\widetilde{\mu}_k(x_k, x_0) = \frac{1}{\sqrt{\alpha_k}} \left(x_k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon\right). \tag{3.27}$$

Proof. From (3.7), we have

$$x_0 = \frac{x_k - \sqrt{1-\bar{\alpha}_k}\epsilon}{\sqrt{\bar{\alpha}_k}}. \tag{3.28}$$

Substituting this expression into (3.15) gives

$$\widetilde{\mu}_k(x_k, x_0) = \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k}\frac{x_k - \sqrt{1-\bar{\alpha}_k}\epsilon}{\sqrt{\bar{\alpha}_k}} + \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}x_k. \tag{3.29}$$

Since $\bar{\alpha}_k = \alpha_k\bar{\alpha}_{k-1}$, the first term becomes

$$\frac{\beta_k}{\sqrt{\alpha_k}(1-\bar{\alpha}_k)}x_k - \frac{\beta_k}{\sqrt{\alpha_k}\sqrt{1-\bar{\alpha}_k}}\epsilon. \tag{3.30}$$

The coefficient of x_k is therefore

$$\frac{\beta_k}{\sqrt{\alpha_k}(1-\bar{\alpha}_k)} + \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k} = \frac{\beta_k + \alpha_k(1-\bar{\alpha}_{k-1})}{\sqrt{\alpha_k}(1-\bar{\alpha}_k)} \tag{3.31}$$

$$= \frac{1-\alpha_k\bar{\alpha}_{k-1}}{\sqrt{\alpha_k}(1-\bar{\alpha}_k)} = \frac{1}{\sqrt{\alpha_k}}. \tag{3.32}$$

Combining the x_k and ϵ terms yields (3.27). □

The preceding proposition gives a direct reason to predict the noise. If a neural network $\epsilon_\theta(x_k, k)$ estimates the noise component ϵ in x_k, then it also determines an estimate of the reverse posterior mean:

$$\mu_\theta(x_k, k) = \frac{1}{\sqrt{\alpha_k}}\left(x_k - \frac{\beta_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta(x_k, k)\right). \tag{3.33}$$

Motivated by the Gaussian oracle posterior in Lemma 3.3, diffusion models introduce a learned reverse Markov chain

$$p_\theta(x_0, \ldots, x_{K-1} \mid x_K) = \prod_{k=1}^{K} p_\theta(x_{k-1} \mid x_k), \tag{3.34}$$

with Gaussian transitions

$$p_\theta(x_{k-1} \mid x_k) = \mathcal{N}(x_{k-1}; \mu_\theta(x_k, k), \sigma_k^2 I). \tag{3.35}$$

The mean μ_θ is parameterized through (3.33). The variance σ_k^2 may be fixed or learned; in the simplified formulation below, it is treated as fixed.

We now derive the connection between mean learning and noise prediction. For two Gaussians

$$q(x_{k-1} \mid x_k, x_0) = \mathcal{N}(\widetilde{\mu}_k, \widetilde{\beta}_k I), \qquad p_\theta(x_{k-1} \mid x_k) = \mathcal{N}(\mu_\theta, \sigma_k^2 I), \tag{3.36}$$

the Gaussian KL divergence is

$$D_{\mathrm{KL}}\left(q(x_{k-1} \mid x_k, x_0) \,\|\, p_\theta(x_{k-1} \mid x_k)\right) \tag{3.37}$$

$$= \frac{1}{2}\left[d\log\frac{\sigma_k^2}{\widetilde{\beta}_k} - d + \frac{d\widetilde{\beta}_k}{\sigma_k^2} + \frac{1}{\sigma_k^2}\left\|\widetilde{\mu}_k(x_k, x_0) - \mu_\theta(x_k, k)\right\|_2^2\right]. \tag{3.38}$$

When σ_k^2 is fixed, the only term depending on θ is the mean-matching term. Hence minimizing this KL divergence with respect to θ is equivalent to minimizing

$$\|\widetilde{\mu}_k(x_k, x_0) - \mu_\theta(x_k, k)\|_2^2 . \tag{3.39}$$

Using (3.27) and (3.33), this term becomes

$$\|\widetilde{\mu}_k(x_k, x_0) - \mu_\theta(x_k, k)\|_2^2 = \frac{\beta_k^2}{\alpha_k(1-\bar{\alpha}_k)} \|\epsilon - \epsilon_\theta(x_k, k)\|_2^2 . \tag{3.40}$$

Thus, up to a diffusion-step-dependent weight, learning the reverse mean is equivalent to predicting the injected noise.

This motivates the practical denoising objective. One samples

$$x_0 \sim p_{\text{data}}, \qquad k \sim \text{Unif}\{1, \dots, K\}, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{3.41}$$

constructs

$$x_k = \sqrt{\bar{\alpha}_k} x_0 + \sqrt{1-\bar{\alpha}_k}\epsilon, \tag{3.42}$$

and minimizes the simplified noise-prediction loss

$$\mathcal{L}_{\text{DM}}(\theta) = \mathbb{E}_{x_0,k,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_k, k)\|_2^2\right]. \tag{3.43}$$

Compared with the weighted objective suggested by (3.40), the simplified objective drops the diffusion-step-dependent weights and is commonly used in practice [Ho+20].

Algorithm 4 Training a denoising diffusion model

Require: Dataset $\mathcal{D} = \{x_0^i\}_{i=1}^N$, diffusion horizon K, variance schedule $\{\beta_k\}_{k=1}^K$, denoising network ϵ_θ, learning rate η

1: Compute $\alpha_k = 1 - \beta_k$ and $\bar{\alpha}_k = \prod_{i=1}^k \alpha_i$ for $k = 1, \dots, K$
2: **while** not converged **do**
3: Sample a minibatch $x_0 \sim \mathcal{D}$
4: Sample diffusion indices $k \sim \text{Unif}\{1, \dots, K\}$
5: Sample noise $\epsilon \sim \mathcal{N}(0, I)$
6: Construct noisy samples

$$x_k = \sqrt{\bar{\alpha}_k} x_0 + \sqrt{1-\bar{\alpha}_k}\epsilon$$

7: Compute the denoising loss

$$\widehat{\mathcal{L}}_{\text{DM}}(\theta) = \|\epsilon - \epsilon_\theta(x_k, k)\|_2^2$$

8: Update parameters

$$\theta \leftarrow \theta - \eta \nabla_\theta \widehat{\mathcal{L}}_{\text{DM}}(\theta)$$

9: **end while**
10: **return** trained denoising network ϵ_θ

Remark 3.5 (Training as supervised denoising). Algorithm 4 shows that diffusion-model training can be implemented as a supervised regression problem. The target is the injected Gaussian noise ϵ, and the input is the corrupted sample x_k together with the diffusion index k. The closed-form marginal in Lemma 3.2 is what makes this training procedure simple.

After training, generation starts from

$$x_K \sim \mathcal{N}(0, I) \tag{3.44}$$

and applies the learned reverse transitions for $k = K, K-1, \ldots, 1$. In the noise-prediction parameterization, a typical sampling step is

$$x_{k-1} = \mu_\theta(x_k, k) + \sigma_k \xi_k, \qquad \xi_k \sim \mathcal{N}(0, I), \tag{3.45}$$

with ξ_1 often set to zero in the final step. The resulting x_0 is the generated sample.

3.2 Conditional Diffusion Models as Stochastic Policies

We follow Janner et al. [Jan+22] and Ajay et al. [Aja+23] for the use of diffusion models in sequential decision-making, and Chi et al. [Chi+25] for the action-diffusion viewpoint of Diffusion Policy. The diffusion model in Section 3.1 defines an unconditional distribution over clean samples x_0. For decision-making, however, the generated object should depend on the current state, observation, goal, or a latent representation of the environment. We denote such conditioning information by c.

A conditional diffusion model keeps the forward noising process unchanged and conditions only the reverse denoising process on c. Thus, the same forward identity from Lemma 3.2 still applies:

$$x_k = \sqrt{\bar{\alpha}_k} x_0 + \sqrt{1 - \bar{\alpha}_k}\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{3.46}$$

The difference is that the denoising network now receives the conditioning variable as an additional input.

Definition 3.6 (Conditional diffusion model)**.** Let c be a conditioning variable. A conditional diffusion model is a reverse denoising Markov chain

$$p_\theta(x_0, \ldots, x_{K-1} \mid x_K, c) = \prod_{k=1}^{K} p_\theta(x_{k-1} \mid x_k, c, k), \tag{3.47}$$

where each reverse transition is parameterized as

$$p_\theta(x_{k-1} \mid x_k, c, k) = \mathcal{N}(x_{k-1}; \mu_\theta(x_k, c, k), \sigma_k^2 I). \tag{3.48}$$

In the noise-prediction parameterization, the mean is obtained from a conditional noise predictor $\epsilon_\theta(x_k, c, k)$ by the same formula as in (3.33), namely

$$\mu_\theta(x_k, c, k) = \frac{1}{\sqrt{\alpha_k}} \left(x_k - \frac{\beta_k}{\sqrt{1 - \bar{\alpha}_k}} \epsilon_\theta(x_k, c, k) \right). \tag{3.49}$$

The conditional training objective is the direct analogue of (3.43):

$$\mathcal{L}_{\mathrm{CDM}}(\theta) = \mathbb{E}_{x_0, c, k, \epsilon} \left[\|\epsilon - \epsilon_\theta(x_k, c, k)\|_2^2 \right], \tag{3.50}$$

where x_k is constructed by (3.46). Thus, conditioning changes the denoising network but not the Gaussian corruption process. The same training procedure as in Algorithm 4 can be used after replacing $\epsilon_\theta(x_k, k)$ by $\epsilon_\theta(x_k, c, k)$.

The learned reverse chain induces a conditional distribution over clean samples:

$$p_\theta(x_0 \mid c) = \int p(x_K) \prod_{k=1}^{K} p_\theta(x_{k-1} \mid x_k, c, k)\, dx_1 \cdots dx_K, \tag{3.51}$$

where $p(x_K) = \mathcal{N}(0, I)$. Although this marginal density is usually not evaluated explicitly, it specifies the distribution sampled by the conditional reverse denoising process.

We now interpret this conditional distribution as a stochastic policy. Let $\mathcal{A}$ be the action space. For a horizon $H \geq 1$, define the action object

$$x_0^t = (x_{0,1}^t, \ldots, x_{0,H}^t) \in \mathcal{A}^H. \tag{3.52}$$

When $H = 1$, the generated object is a single action. When $H > 1$, it is an action chunk. Action chunks and receding-horizon execution are central components of Diffusion Policy [Chi+25]. In continuous control, if $\mathcal{A} \subseteq \mathbb{R}^{d_a}$, then $x_0^t \in \mathbb{R}^{Hd_a}$.

Let z_t be a latent representation of the environment state s_t, for example

$$z_t = \phi(s_t), \tag{3.53}$$

where ϕ is an observation encoder. In crowd navigation, z_t is the encoded crowd observation. Taking $c = z_t$ and taking the clean sample to be the action object x_0^t, the conditional diffusion model induces

$$\pi_\theta(x_0^t \mid z_t) = \int p(x_K^t) \prod_{k=1}^{K} p_\theta(x_{k-1}^t \mid x_k^t, z_t, k)\, dx_1^t \cdots dx_K^t. \tag{3.54}$$

This is a conditional distribution over actions or action chunks, and therefore fits the stochastic-policy viewpoint of Definition 2.3.

The following definition restates the action-diffusion viewpoint of Diffusion Policy in the stochastic-policy notation introduced in Chapter 2.

Definition 3.7 (Diffusion policy)**.** Let z be a state or observation representation, and let x_0 be an action object in $\mathcal{A}^H$. A diffusion policy is a stochastic policy whose conditional distribution $\pi_\theta(\cdot \mid z)$ is induced by a conditional reverse diffusion process:

$$x_K \sim \mathcal{N}(0, I), \qquad x_{k-1} \sim p_\theta(\cdot \mid x_k, z, k), \qquad k = K, \ldots, 1. \tag{3.55}$$

Equivalently, the induced conditional distribution is given by (3.54).

3.3 Expressiveness Beyond Gaussian Policies

We follow the multimodal action-distribution motivation of Chi et al. [Chi+25] in explaining why diffusion policies are useful for continuous control. For the classical limitation of single Gaussian densities in multimodal conditional modeling, we refer to the mixture-density perspective of Bishop [Bis94]. The propositions below give a self-contained illustration in the stochastic-policy notation used throughout this thesis.

The key point is expressiveness. A diffusion policy is not restricted to a single Gaussian mode; instead, it defines a flexible non-Gaussian conditional distribution over the generated action object. For a single-step policy, this object is $x_0 = a \in \mathcal{A}$. For an action-chunk policy with horizon H, it is

$$x_0 = (a_1, \ldots, a_H) \in \mathcal{A}^H. \tag{3.56}$$

In continuous control, we identify x_0 with a vector in $\mathbb{R}^m$, where $m = d_a$ for single-step actions and $m = Hd_a$ for action chunks.

A common policy class is the Gaussian action-object policy

$$\pi_G(x_0 \mid z) = \mathcal{N}(x_0; \mu_G(z), \Sigma_G(z)), \qquad x_0 \in \mathbb{R}^m. \tag{3.57}$$

If $\Sigma_G(z)$ is a full covariance matrix, then this policy can represent correlations between different components of an action chunk. Thus, the limitation of a Gaussian chunk policy is not the absence of temporal dependence, but the fact that the joint distribution over the whole action object remains one Gaussian mode.

Proposition 3.8 (Ellipsoidal geometry of Gaussian policies)**.** *Let*

$$\pi_G(x_0 \mid z) = \mathcal{N}(x_0; \mu_G(z), \Sigma_G(z)), \tag{3.58}$$

where $\Sigma_G(z)$ is positive definite. For each fixed z, the density $\pi_G(\cdot \mid z)$ has a unique mode at $\mu_G(z)$, and its upper level sets are ellipsoids:

$$\{x_0 : \pi_G(x_0 \mid z) \geq c\} = \left\{x_0 : (x_0 - \mu_G(z))^\top \Sigma_G(z)^{-1}(x_0 - \mu_G(z)) \leq r_c\right\} \tag{3.59}$$

for a suitable constant r_c.

Proof. For fixed z, the Gaussian density is proportional to

$$\exp\left(-\frac{1}{2}(x_0 - \mu_G(z))^\top \Sigma_G(z)^{-1}(x_0 - \mu_G(z))\right). \tag{3.60}$$

The exponent is maximized uniquely at $x_0 = \mu_G(z)$. Moreover, $\pi_G(x_0 \mid z) \geq c$ is equivalent to an upper bound on the corresponding quadratic form, giving the stated ellipsoid. □

The ellipsoidal geometry means that a Gaussian policy can stretch or rotate one high-probability region, but cannot create several separated high-probability regions. This becomes restrictive when several qualitatively different actions or local plans are reasonable under the same observation. In crowd navigation, for example, passing a pedestrian on the left and passing on the right may both be valid maneuvers.

The following example makes this representation gap explicit in KL divergence.

Proposition 3.9 (Gaussian KL gap for a two-mode action distribution)**.** *Fix a latent state z and suppose the desired conditional action-object distribution is*

$$P^\star(\cdot \mid z) = \frac{1}{2}\mathcal{N}(-u, \sigma^2 I_m) + \frac{1}{2}\mathcal{N}(u, \sigma^2 I_m), \tag{3.61}$$

where $u \in \mathbb{R}^m$ and $\sigma > 0$. Then the best single Gaussian approximation in forward KL satisfies

$$\inf_{Q \text{ Gaussian}} D_{\mathrm{KL}}(P^\star(\cdot \mid z) \| Q) \geq \max\left\{0, \frac{1}{2}\log\left(1 + \frac{\|u\|_2^2}{\sigma^2}\right) - \log 2\right\}. \tag{3.62}$$

In particular, if $u \neq 0$, no single Gaussian distribution equals $P^\star(\cdot \mid z)$.

Proof. Write $P = P^\star(\cdot \mid z)$. The mean of P is zero and its covariance is

$$\Sigma_P = \sigma^2 I_m + uu^\top. \tag{3.63}$$

Let $Q = \mathcal{N}(\mu, \Sigma)$ with density q. Since

$$D_{\mathrm{KL}}(P \| Q) = -h(P) - \mathbb{E}_P[\log q(X)], \tag{3.64}$$

and

$$-\log q(X) = \frac{m}{2}\log(2\pi) + \frac{1}{2}\log\det\Sigma + \frac{1}{2}(X - \mu)^\top \Sigma^{-1}(X - \mu), \tag{3.65}$$

the expectation is minimized by matching the mean and covariance of P. Therefore the best Gaussian is

$$Q^\star = \mathcal{N}(0, \Sigma_P). \tag{3.66}$$

At this optimum,

$$\inf_{Q \text{ Gaussian}} D_{\mathrm{KL}}(P\|Q) = D_{\mathrm{KL}}(P\|Q^\star) = h(Q^\star) - h(P). \tag{3.67}$$

Let $C \in \{-1, +1\}$ denote the mixture component. Then

$$h(P) = h(X) \le h(X, C) = h(C) + h(X \mid C) = \log 2 + \frac{m}{2}\log(2\pi e\sigma^2). \tag{3.68}$$

On the other hand, by the matrix determinant lemma,

$$h(Q^\star) = \frac{1}{2}\log\left((2\pi e)^m \det(\sigma^2 I_m + uu^\top)\right) \tag{3.69}$$

$$= \frac{m}{2}\log(2\pi e\sigma^2) + \frac{1}{2}\log\left(1 + \frac{\|u\|_2^2}{\sigma^2}\right). \tag{3.70}$$

Combining (3.67), (3.68), and (3.70) gives

$$D_{\mathrm{KL}}(P\|Q^\star) \ge \frac{1}{2}\log\left(1 + \frac{\|u\|_2^2}{\sigma^2}\right) - \log 2. \tag{3.71}$$

Together with the non-negativity of KL divergence, this proves the stated lower bound. Since a genuine two-mode Gaussian mixture is not itself Gaussian when $u \neq 0$, the representation error cannot be removed within the single Gaussian policy class. □

We now show that the same two-mode distribution can be represented exactly by an idealized diffusion policy. Thus, the best single Gaussian policy has a KL gap, whereas a diffusion-style reverse denoising policy can have zero KL error.

Proposition 3.10 (Exact representation by a one-step diffusion policy)**.** *Fix the target distribution*

$$P^\star(\cdot \mid z) = \frac{1}{2}\mathcal{N}(-u, \sigma^2 I_m) + \frac{1}{2}\mathcal{N}(u, \sigma^2 I_m). \tag{3.72}$$

There exists a one-step reverse denoising policy whose induced distribution is exactly $P^\star(\cdot \mid z)$. Consequently,

$$D_{\mathrm{KL}}\big(P^\star(\cdot \mid z) \,\|\, \pi_\theta(\cdot \mid z)\big) = 0 \tag{3.73}$$

for this policy.

Proof. Consider a one-step reverse chain with $K = 1$. Sample

$$x_1 \sim \mathcal{N}(0, I_m). \tag{3.74}$$

Define two regions of the noise space:

$$B_+ = \{x_1 \in \mathbb{R}^m : (x_1)_1 \ge 0\}, \qquad B_- = \{x_1 \in \mathbb{R}^m : (x_1)_1 < 0\}. \tag{3.75}$$

By symmetry,

$$\mathbb{P}(x_1 \in B_+) = \mathbb{P}(x_1 \in B_-) = \frac{1}{2}. \tag{3.76}$$

Define the reverse transition

$$p_\theta(x_0 \mid x_1, z) = \begin{cases} \mathcal{N}(x_0; u, \sigma^2 I_m), & x_1 \in B_+, \\ \mathcal{N}(x_0; -u, \sigma^2 I_m), & x_1 \in B_-. \end{cases} \tag{3.77}$$

Marginalizing over x_1 gives

$$\pi_\theta(x_0 \mid z) = \int p(x_1) p_\theta(x_0 \mid x_1, z)\, dx_1 \tag{3.78}$$

$$= \frac{1}{2}\mathcal{N}(x_0; u, \sigma^2 I_m) + \frac{1}{2}\mathcal{N}(x_0; -u, \sigma^2 I_m) = P^\star(x_0 \mid z). \tag{3.79}$$

Hence the induced policy distribution equals the target distribution exactly, and the KL divergence is zero. □

The example above is stated for a general action object $x_0 \in \mathbb{R}^m$. When $H = 1$, the two modes correspond to two different single-step actions. When $H > 1$, the same construction applies to action chunks $x_0 = (a_1, \ldots, a_H) \in \mathcal{A}^H$, where the two modes can represent two different short-horizon local plans.

Corollary 3.11 (Action chunks amplify the role of expressiveness)**.** *Let $x_0 \in \mathcal{A}^H$ be an action chunk. A full-covariance Gaussian policy on $\mathcal{A}^H$ can represent correlations between the actions in the chunk, but it remains a single Gaussian mode in the joint action-object space. If the desired short-horizon behavior contains several distinct local-plan modes, a non-Gaussian policy class such as a diffusion policy can represent this structure more naturally.*

In crowd navigation, an action chunk can be interpreted as a short-horizon local plan. The main advantage of a diffusion action-chunk policy is therefore not merely that it outputs several future actions, but that it can represent multi-modal distributions over local plans, such as passing on different sides of a pedestrian or choosing between moving forward and slowing down.

Chapter 4

Robot Crowd Navigation Background

This chapter introduces the robot crowd-navigation background needed for the method and experiments. The discussion follows the main methodological transition from rule-based velocity-space collision avoidance to learning-based interaction representation. We use ORCA as the representative rule-based method and CrowdNav++ as the representative learning-based method. This choice also matches the experimental design in Chapter 6, where ORCA and CrowdNav++ serve as the two main baselines.

We first fix the notation for crowd navigation and relate it to the MDP formalism from Chapter 2. We then describe ORCA, which computes safe velocities through reciprocal constraints in velocity space. Next, we present encoder-based reinforcement learning for crowd navigation and introduce the main interaction encoders used in this line of work. The chapter concludes with CrowdNav++ and uses it to clarify the distinction between improving state representation and improving action generation.

4.1 Crowd Navigation and Common Notation

Robot crowd navigation considers a robot moving toward a goal in an environment shared with multiple pedestrians. The robot must make progress toward the goal while avoiding collisions and maintaining sufficient clearance from nearby humans. These requirements may favor different motions. As illustrated in Figure 4.1, several candidate routes can connect the robot's current position to the goal. A shorter route may reduce travel time but pass through a crowded region, whereas a longer detour may provide greater clearance and more time to respond to pedestrian motion. The navigation problem therefore requires a balance between efficiency and safety, together with continual adaptation to the evolving crowd.

Following the MDP notation introduced in Definition 2.1, we model crowd navigation as an episodic continuous-control problem. At environment time t, the robot observes the current crowd state, selects a velocity command, receives a reward, and transitions to the next state. The action is written as

$$a_t = (v_{x,t}, v_{y,t}) \in \mathcal{A} \subseteq \mathbb{R}^2, \tag{4.1}$$

where $v_{x,t}$ and $v_{y,t}$ are the commanded planar velocity components.

The crowd-navigation state is denoted by

$$s_t = (w_t, h_1^t, \ldots, h_{n_t}^t), \tag{4.2}$$

Figure 4.1: Schematic illustration of robot crowd navigation. The robot moves toward its goal while accounting for nearby pedestrians and their possible motion. The candidate routes illustrate the trade-off between navigation efficiency and safety: a shorter route may pass through a more crowded region, whereas a longer detour may provide greater clearance. The illustration was generated with ChatGPT and is included only to provide visual intuition.

where w_t is the robot state and n_t is the number of visible humans. The robot state contains its position, velocity, goal-related information, radius, and maximum speed. For each visible human i, the feature h_i^t contains the information available to the robot about that pedestrian.

In the prediction-aware setting used later in the thesis, the human feature is written as

$$h_i^t = \left(u_i^t, \widehat{u}_i^{t+1:t+L}\right), \tag{4.3}$$

where u_i^t denotes the current human state and $\widehat{u}_i^{t+1:t+L}$ denotes predicted future states over a horizon of length L. When future predictions are not used, the feature reduces to $h_i^t = u_i^t$.

Throughout the thesis, t denotes environment time. The index k is reserved for diffusion time in the reverse denoising process, as used in Chapter 3 and Chapter 5. The index ℓ denotes an offset in the human-prediction horizon, for example $\widehat{u}_i^{t+\ell}$ for $\ell = 1, \ldots, L$. The horizon of a robot action chunk is denoted by H.

By Definition 2.3, a stochastic crowd-navigation policy assigns a conditional distribution over velocity commands:

$$a_t \sim \pi_\theta(\cdot \mid s_t). \tag{4.4}$$

Because the number of visible humans can vary over time, learning-based methods typically map the crowd state to a fixed-dimensional latent representation,

$$z_t = E_\psi(s_t), \tag{4.5}$$

and condition the policy on this representation:

$$a_t \sim \pi_\theta(\cdot \mid z_t). \tag{4.6}$$

The encoder E_ψ must therefore summarize a variable-size crowd observation while preserving the information relevant to navigation.

In particular, the human set $\{h_i^t\}_{i=1}^{n_t}$ is unordered, so the encoded representation should not depend on an arbitrary ordering of pedestrians. It should also capture how the current and predicted motion of nearby humans constrains the robot's available routes. Pedestrian motion is uncertain, and interactions among humans can change which regions remain safe or efficient. These properties motivate the interaction-aware crowd encoders introduced in the following sections.

4.2 Rule-Based Collision Avoidance: ORCA

Before learning-based methods became standard, multi-agent collision avoidance was often approached through hand-designed local rules. ORCA is a classical velocity-based method in this family [Ber+11]. It is decentralized: each agent chooses its own velocity from local observations of nearby agents, without explicit communication. In this subsection only, the indices i and j denote generic agents rather than specifically humans.

Consider two disk-shaped agents i and j in the plane, with positions $p_i, p_j \in \mathbb{R}^2$, velocities $v_i, v_j \in \mathbb{R}^2$, and radii $r_i, r_j > 0$. Define the relative position and combined radius by

$$p_{ij} = p_j - p_i, \qquad r_{ij} = r_i + r_j. \tag{4.7}$$

For a time horizon $\tau > 0$, the velocity obstacle induced by agent j for agent i is the set of relative velocities that would lead to a collision within time τ:

$$\mathrm{VO}_{ij}^{\tau} = \left\{ v \in \mathbb{R}^2 \;\middle|\; \exists \eta \in [0, \tau] \text{ such that } \eta v \in D(p_{ij}, r_{ij}) \right\}, \tag{4.8}$$

where $D(p, r)$ is the open disk centered at p with radius r. Thus, if the relative velocity $v_i - v_j$ lies in VO_{ij}^{τ}, then the two agents are predicted to collide within the horizon τ if they keep their current velocities.

The ORCA construction moves the relative velocity outside this velocity obstacle. Let u_{ij} be the smallest correction that makes the relative velocity collision-free:

$$u_{ij} = \arg\min_{u \in \mathbb{R}^2} \left\{ \|u\|_2 \;\middle|\; v_i - v_j + u \notin \mathrm{VO}_{ij}^{\tau} \right\}. \tag{4.9}$$

Let n_{ij} be the outward normal to the boundary of the velocity obstacle at the closest collision-free point. The reciprocal assumption of ORCA is that agents i and j share responsibility for avoiding the collision. Therefore, agent i takes half of the required relative-velocity correction. This gives the pairwise ORCA half-plane

$$\mathrm{ORCA}_{ij}^{\tau} = \left\{ v \in \mathbb{R}^2 \;\middle|\; \left(v - \left(v_i + \frac{1}{2} u_{ij} \right) \right)^{\top} n_{ij} \geq 0 \right\}. \tag{4.10}$$

This half-plane contains the velocities for agent i that are safe with respect to agent j, assuming that agent j follows the same reciprocal principle.

For multiple neighboring agents, the feasible velocity set is the intersection of all pairwise ORCA constraints and the agent's speed constraint:

$$\mathcal{V}_i^{\mathrm{ORCA}} = \left\{ v \in \mathbb{R}^2 : \|v\|_2 \leq v_i^{\max} \right\} \cap \bigcap_{j \neq i} \mathrm{ORCA}_{ij}^{\tau}. \tag{4.11}$$

Given a preferred goal-directed velocity v_i^{pref}, ORCA selects the feasible velocity closest to it:

$$v_i^{\mathrm{ORCA}} = \arg\min_{v \in \mathcal{V}_i^{\mathrm{ORCA}}} \left\| v - v_i^{\mathrm{pref}} \right\|_2^2. \tag{4.12}$$

Since the constraints are half-planes in two-dimensional velocity space, this optimization is efficient and interpretable.

ORCA therefore provides a strong local collision-avoidance mechanism. In this thesis it plays two roles: it is a representative rule-based baseline in Chapter 6, and it is used to generate demonstration trajectories for behavioral-cloning pretraining in Chapter 5. At the same time, its decision variable is still a single velocity chosen from a local feasible set. This makes it a natural point of comparison for learning-based policies, which seek to optimize navigation behavior over longer interaction horizons.

4.3 Encoder-Based Reinforcement Learning for Crowd Navigation

Learning-based crowd-navigation methods replace hand-designed velocity rules by policies trained from simulated interaction experience. Their common structure is an encoder-policy pipeline:

$$s_t \xmapsto{E_\psi} z_t \xmapsto{\pi_\theta} a_t. \tag{4.13}$$

The encoder E_ψ maps the robot and crowd state to a fixed-dimensional representation z_t, and the policy head π_θ maps this representation to an action distribution. The actor–critic and PPO machinery introduced in Sections 2.4 and 2.5 can then be used to train the policy.

This viewpoint separates two modeling questions. The first question is how to represent the crowd state:

$$E_\psi : (w_t, h_1^t, \dots, h_{n_t}^t) \mapsto z_t. \tag{4.14}$$

The second question is how to represent the action distribution:

$$\pi_\theta(\cdot \mid z_t). \tag{4.15}$$

This separation will be useful throughout the thesis. The learning-based methods reviewed below mainly develop the encoder E_ψ, while the method introduced in the next chapter keeps the encoder-policy viewpoint but changes the action-generation model.

Definition 4.1 (Encoder-based crowd-navigation policy)**.** An encoder-based crowd-navigation policy consists of a crowd-state encoder E_ψ and an action head π_θ. Given a state

$$s_t = (w_t, h_1^t, \dots, h_{n_t}^t),$$

the encoder computes

$$z_t = E_\psi(s_t), \tag{4.16}$$

and the action is sampled from

$$a_t \sim \pi_\theta(\cdot \mid z_t). \tag{4.17}$$

A basic requirement for E_ψ is permutation invariance with respect to the ordering of humans. The humans are represented as a set, not as a sequence with intrinsic order. Thus, reordering the input list of humans should not change the encoded crowd representation.

Definition 4.2 (Permutation-invariant crowd encoder)**.** A crowd encoder E_ψ is permutation invariant if, for every permutation σ of $\{1, \dots, n_t\}$,

$$E_\psi(w_t, h_1^t, \dots, h_{n_t}^t) = E_\psi(w_t, h_{\sigma(1)}^t, \dots, h_{\sigma(n_t)}^t). \tag{4.18}$$

A common construction is to compute pairwise human–robot embeddings and then aggregate them by a permutation-invariant operation:

$$e_i^t = \phi_\psi(w_t, h_i^t), \qquad c_t = \mathrm{Pool}_{i=1}^{n_t} e_i^t, \qquad z_t = \rho_\psi(w_t, c_t), \tag{4.19}$$

where ϕ_ψ and ρ_ψ are neural networks and Pool may be a sum, mean, max, or attention-based aggregation.

Proposition 4.3 (Permutation invariance of pooling encoders)**.** *If* Pool *is permutation invariant, then the encoder in* (4.19) *is permutation invariant with respect to the ordering of humans.*

Proof. The multiset $\{e_i^t\}_{i=1}^{n_t}$ is unchanged by a permutation of the humans. Since Pool is permutation invariant, c_t is unchanged. Therefore $z_t = \rho_\psi(w_t, c_t)$ is unchanged. □

The development of learning-based crowd navigation can be read through this encoder lens. Early methods use pairwise features and pooling; attention-based methods learn which humans are most relevant; recurrent methods add temporal memory; graph methods model spatial dependencies among agents; and CrowdNav++ combines these tools with short-horizon prediction.

4.4 Interaction Encoders

This section introduces the main neural modules used by learning-based crowd-navigation methods. The aim is to define the tools needed to understand CrowdNav++ and the observation encoder used later in Chapter 5.

Pooling and attention

A simple pooling encoder treats each human independently before aggregation. This is permutation invariant, but it may give equal importance to all humans. In dense crowds, only a subset of pedestrians may be critical for the robot's next decision. Attention-based pooling addresses this by assigning a learned weight to each human.

Given pairwise embeddings

$$e_i^t = \phi_\psi(w_t, h_i^t), \qquad i = 1, \ldots, n_t, \tag{4.20}$$

an attention score is computed as

$$q_i^t = f_{\mathrm{att}}(e_i^t), \tag{4.21}$$

and normalized by

$$\alpha_i^t = \frac{\exp(q_i^t)}{\sum_{j=1}^{n_t} \exp(q_j^t)}. \tag{4.22}$$

The crowd representation is then

$$c_t = \sum_{i=1}^{n_t} \alpha_i^t e_i^t. \tag{4.23}$$

This mechanism captures the main idea behind attention-based crowd-navigation policies such as SARL [Che+19]: the encoder learns which pedestrians are most relevant for the robot's decision.

Recurrent updates

A single observation may not reveal the intention or short-term motion pattern of a pedestrian. Recurrent encoders maintain a hidden state and thereby allow the policy to condition on recent interaction history. This idea is used in structural recurrent models for crowd navigation such as DS-RNN [Liu+21].

Let y_t be a current crowd-interaction feature, for example the output of a pooling or graph encoder. A generic recurrent encoder updates a hidden state b_t by

$$b_t = R_\omega(y_t, b_{t-1}). \tag{4.24}$$

A standard choice is a gated recurrent unit.

Definition 4.4 (GRU update)**.** Given input y_t and previous hidden state b_{t-1}, a GRU computes

$$z_t^{\text{gate}} = \sigma\left(W_z y_t + U_z b_{t-1} + c_z\right), \tag{4.25}$$
$$r_t^{\text{gate}} = \sigma\left(W_r y_t + U_r b_{t-1} + c_r\right), \tag{4.26}$$
$$\widetilde{b}_t = \tanh\left(W_b y_t + U_b\left(r_t^{\text{gate}} \odot b_{t-1}\right) + c_b\right), \tag{4.27}$$
$$b_t = (1 - z_t^{\text{gate}}) \odot b_{t-1} + z_t^{\text{gate}} \odot \widetilde{b}_t. \tag{4.28}$$

Here σ is the logistic sigmoid and $\odot$ denotes componentwise multiplication.

The update gate controls how much the hidden state changes, while the reset gate controls how much past information is used in forming the candidate hidden state. In crowd navigation, the recurrent hidden state can summarize recent relative motion and short-term interaction history:

$$y_t = E_\psi(s_t), \qquad b_t = \text{GRU}(y_t, b_{t-1}), \qquad a_t \sim \pi_\theta(\cdot \mid b_t). \tag{4.29}$$

Graph message passing

Another way to improve the crowd-state encoder is to represent the agents as a graph. The robot and humans are nodes, and edges represent spatial or interaction relationships. This viewpoint is used by relational graph learning methods for crowd navigation [Che+20].

Let

$$\mathcal{V}_t = \{r, 1, \ldots, n_t\} \tag{4.30}$$

be the node set, where r denotes the robot and $i = 1, \ldots, n_t$ denote humans. A graph encoder initializes node features by

$$x_{r,t}^{(0)} = f_r(w_t), \qquad x_{i,t}^{(0)} = f_h(h_i^t), \quad i = 1, \ldots, n_t. \tag{4.31}$$

A generic message-passing layer updates each node by aggregating information from its neighbors:

$$m_{i,t}^{(\ell)} = \sum_{j \in \mathcal{N}_t(i)} \alpha_{ij,t}^{(\ell)} W^{(\ell)} x_{j,t}^{(\ell)}, \tag{4.32}$$
$$x_{i,t}^{(\ell+1)} = \sigma\left(W_0^{(\ell)} x_{i,t}^{(\ell)} + m_{i,t}^{(\ell)}\right). \tag{4.33}$$

Here $\mathcal{N}_t(i)$ is the neighborhood of node i, and $\alpha_{ij,t}^{(\ell)}$ may be fixed graph weights or learned attention weights.

Graph encoders are useful because human–human interactions can influence the robot indirectly. For example, one pedestrian may constrain the motion of another pedestrian, thereby changing the free space available to the robot. A graph representation allows the encoder to capture such multi-agent dependencies beyond independent human–robot pairwise features.

Scaled dot-product attention

Attention-based graph models are often expressed in query-key-value form. Given matrices Q, K, V, scaled dot-product attention is

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V, \tag{4.34}$$

where d is the key dimension. Multi-head attention applies this operation in several learned subspaces and concatenates the outputs. In crowd navigation, this mechanism can be used to aggregate human features, model human–human interactions, or let the robot attend to interaction-aware human embeddings.

The modules above describe the main representation tools behind the learning-based line of crowd navigation. CrowdNav++ combines them in a prediction-aware architecture.

4.5 CrowdNav++ as a Strong Learning-Based Baseline

We now describe CrowdNav++ [Liu+23], the main learning-based baseline used in this thesis. CrowdNav++ follows the encoder-policy view: it uses model-free reinforcement learning, but places substantial modeling capacity in the state encoder. In particular, the encoder combines short-horizon human motion prediction, attention-based interaction modeling, and temporal recurrence. The notation below abstracts the main architectural components rather than reproducing implementation-level tensor shapes.

Prediction-augmented state

CrowdNav++ augments the current human observations with predicted future human motion. Let the recent observed history of human i be

$$u_i^{t-M:t} = (u_i^{t-M}, \ldots, u_i^t). \tag{4.35}$$

A trajectory predictor produces future states over a short horizon:

$$\widehat{u}_i^{t+1:t+L} = \mathrm{Pred}_\eta\left(u_i^{t-M:t}\right), \qquad i = 1, \ldots, n_t. \tag{4.36}$$

The augmented human feature is

$$h_i^t = (u_i^t, \widehat{u}_i^{t+1:t+L}), \tag{4.37}$$

and the policy input is

$$s_t^{++} = (w_t, h_1^t, \ldots, h_{n_t}^t). \tag{4.38}$$

The exact predictor architecture is not essential for the present discussion. What matters is that the navigation policy receives information about likely short-horizon human motion, rather than only the current human states.

Prediction-aware reward

CrowdNav++ also incorporates predicted human motion into its reward [Liu+23]. For each predicted state $\widehat{u}_i^{t+\ell}$, let

$$\mathcal{Z}_i^{t+\ell} = \left\{ p \in \mathbb{R}^2 : \|p - \widehat{p}_i^{t+\ell}\|_2 \le d_{\text{safe}} \right\}, \qquad \ell = 1, \dots, L, \tag{4.39}$$

where $\widehat{p}_i^{t+\ell}$ is the predicted human position. Define

$$I_i^{t+\ell} = \mathbf{1}\left\{ p_t^r \in \mathcal{Z}_i^{t+\ell} \right\}, \tag{4.40}$$

which indicates whether the robot intrudes into the predicted zone of human i at prediction step ℓ. With collision penalty $r_c = -20$, the prediction penalty for human i and the aggregate prediction penalty are

$$r_{\text{pred},i}(s_t) = \min_{\ell=1,\dots,L} \left(I_i^{t+\ell} \frac{r_c}{2^\ell} \right), \tag{4.41}$$

$$r_{\text{pred}}(s_t) = \min_{i=1,\dots,n_t} r_{\text{pred},i}(s_t). \tag{4.42}$$

The factor $2^{-\ell}$ assigns a smaller penalty to intrusions that occur further into the predicted future.

CrowdNav++ additionally uses the potential-based progress reward

$$r_{\text{pot}}(s_t) = 2\left(d_{\text{goal}}^{t-1} - d_{\text{goal}}^{t} \right), \tag{4.43}$$

where d_{goal}^t is the Euclidean distance from the robot to its goal at time t. Let $\mathcal{S}_{\text{goal}}$ denote the goal states and $\mathcal{S}_{\text{fail}}$ the collision states. The complete reward is

$$r(s_t, a_t) = \begin{cases} 10, & s_t \in \mathcal{S}_{\text{goal}}, \\ r_c, & s_t \in \mathcal{S}_{\text{fail}}, \\ r_{\text{pot}}(s_t) + r_{\text{pred}}(s_t), & \text{otherwise}. \end{cases} \tag{4.44}$$

Thus, trajectory prediction affects both the policy input and the reward used for model-free reinforcement learning.

Prediction-aware interaction encoder

The prediction-augmented state s_t^{++} still contains a variable-size set of human features. CrowdNav++ processes this set with attention modules that separate two types of interaction: human–human interaction among pedestrians and robot–human interaction from the perspective of the robot.

Stack the augmented human features into a matrix

$$H_t^{(0)} = \begin{bmatrix} (h_1^t)^\top \\ \vdots \\ (h_{n_t}^t)^\top \end{bmatrix}. \tag{4.45}$$

A human–human attention module computes interaction-aware human embeddings:

$$H_t^{\text{HH}} = \text{MHA}_{\text{HH}}\left(H_t^{(0)} \right), \tag{4.46}$$

where MHA_{HH} denotes multi-head attention over the human features. This step allows each human representation to incorporate information from other pedestrians.

The robot state is embedded as

$$e_t^r = f_r(w_t). \tag{4.47}$$

A robot–human attention module then aggregates the human embeddings from the robot's perspective:

$$c_t^{\mathrm{RH}} = \mathrm{MHA}_{\mathrm{RH}}\left(q_r(e_t^r), K_h(H_t^{\mathrm{HH}}), V_h(H_t^{\mathrm{HH}})\right), \tag{4.48}$$

where q_r, K_h, and V_h are learned query, key, and value maps. The resulting vector c_t^{RH} is a robot-centered summary of the predicted and interaction-aware crowd state.

This description keeps only the role of the interaction structure that is used later in the thesis: CrowdNav++ builds a strong crowd representation before passing it to an actor–critic policy. The detailed graph construction is not needed for the PDPO formulation.

Temporal recurrence and actor–critic output

CrowdNav++ maintains temporal information using a recurrent module. The current robot-centered interaction feature is

$$y_t = [e_t^r, c_t^{\mathrm{RH}}]. \tag{4.49}$$

A recurrent hidden state is updated by

$$b_t = \mathrm{GRU}(y_t, b_{t-1}). \tag{4.50}$$

The hidden state b_t summarizes the current crowd representation and recent interaction history. The actor and critic are computed as

$$\pi_\theta(\cdot \mid s_t^{++}) = f_\pi(b_t), \qquad V_\varphi(s_t^{++}) = f_V(b_t). \tag{4.51}$$

The policy and value function are trained using PPO. Thus, CrowdNav++ can be viewed as an actor–critic method with a prediction-aware spatio-temporal interaction encoder.

The representation pipeline can be summarized as

$$s_t^{++} \longmapsto \{h_i^t\}_{i=1}^{n_t} \longmapsto H_t^{\mathrm{HH}} \longmapsto c_t^{\mathrm{RH}} \longmapsto b_t \longmapsto \pi_\theta(\cdot \mid s_t^{++}). \tag{4.52}$$

CrowdNav++ is therefore a strong representative of the learning-based state-representation direction: it improves how the robot encodes predicted human motion, crowd interaction, and temporal context. By contrast, the method developed in this thesis focuses on the action-generation side, replacing a single-step action policy by a diffusion policy over short-horizon velocity chunks.

4.6 From State Representation to Action Generation

The development reviewed in this chapter can be summarized as a shift from hand-designed velocity selection to learned interaction representation. ORCA constructs a feasible set of locally safe velocities and selects the velocity closest to a preferred direction. Learning-based methods instead learn an encoder-policy pipeline

$$s_t \xmapsto{E_\psi} z_t \xmapsto{\pi_\theta} a_t. \tag{4.53}$$

Within this pipeline, much of the progress has come from improving E_ψ: attention pooling identifies relevant pedestrians, recurrent updates encode temporal history, graph encoders

represent spatial dependencies, and CrowdNav++ combines these components with short-horizon human prediction.

This perspective also motivates the choice of baselines in Chapter 6. ORCA represents the classical rule-based velocity-space approach, while CrowdNav++ represents a strong learning-based state-representation approach. The comparison in Chapter 6 therefore evaluates whether changing the policy's action-generation model provides an additional benefit beyond these two established directions.

The next chapter takes this complementary direction. Instead of mapping z_t directly to a single velocity, PDPO uses a diffusion policy to generate an action chunk,

$$x_0^t = (x_{0,1}^t, \ldots, x_{0,H}^t) \in \mathbb{R}^{Hd_a}, \qquad d_a = 2. \tag{4.54}$$

The executed environment action is the first command,

$$a_t = x_{0,1}^t, \tag{4.55}$$

and a new chunk is generated at the next timestep. Thus, the policy represents a distribution over short-horizon local plans while preserving closed-loop feedback.

This action-generation viewpoint is motivated by the multimodality of dense crowd navigation. Under the same observation, passing on the left, passing on the right, slowing down, or taking a short detour may all be feasible. These alternatives are better viewed as different local plan modes than as small perturbations of a single velocity. The next chapter formulates this idea as Planning Diffusion Policy Optimization.

Chapter 5

Planning Diffusion Policy Optimization for Crowd Navigation

This chapter presents Planning Diffusion Policy Optimization (PDPO) for robot crowd navigation. The method uses a conditional diffusion policy to generate short-horizon velocity chunks, which are executed in a receding-horizon manner: at each environment step, the policy samples a local plan, executes only its first velocity command, and replans from the next crowd observation.

The optimization framework follows the augmented denoising-MDP formulation of DPPO [Ren+25]. In this view, the original crowd-navigation MDP forms the outer decision process, while the reverse denoising chain forms an inner process that generates the action chunk. This makes it possible to apply policy-gradient updates to the tractable denoising transition densities, without evaluating the intractable marginal density of the final action chunk.

The contribution of this chapter is to instantiate and adapt this formulation for robot crowd navigation. We define the crowd-conditioned action-chunk policy and the induced receding-horizon environment policy. We then introduce an offline denoising pretraining stage based on expert action chunks. Finally, we couple the outer environment process and the inner denoising process, make the inherited reward structure explicit, derive the corresponding score decomposition, and formulate the PPO fine-tuning objective used in the online stage.

5.1 Environment MDP and Diffusion Policy Specialization

We use the crowd-navigation notation introduced in Section 4.1. The environment is modeled as Markov decision process

$$\mathcal{M}_{\text{env}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_0, \mathcal{P}, \mathcal{R}, \gamma), \tag{5.1}$$

where $\mathcal{S}$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^2$ is the continuous velocity-action space, $\mathcal{P}_0$ is the initial-state distribution, $\mathcal{P}$ is the transition kernel, $\mathcal{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. At environment time t, the state is

$$s_t = (w_t, h_1^t, \ldots, h_{n_t}^t), \qquad h_i^t = (u_i^t, \widehat{u}_i^{t+1:t+L}), \tag{5.2}$$

where w_t is the robot state, u_i^t is the current state of human i, and $\widehat{u}_i^{t+1:t+L}$ denotes predicted future human states over a short horizon L. In the experiments, these predictions are produced by a constant-velocity predictor. The number of visible humans n_t may vary over time.

The robot executes a two-dimensional velocity command

$$a_t \in \mathcal{A} \subseteq \mathbb{R}^2. \tag{5.3}$$

After executing a_t, the environment transitions according to

$$s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t), \tag{5.4}$$

and returns reward

$$r_t = \mathcal{R}(s_t, a_t). \tag{5.5}$$

Since s_t contains a variable number of humans, PDPO maps it to a fixed-dimensional latent representation:

$$z_t = E_\theta(s_t) \in \mathbb{R}^{d_z}. \tag{5.6}$$

Here E_θ denotes the observation encoder. In the notation of the previous chapter, this corresponds to the encoder E_ψ. In PDPO, the encoder is trained jointly with the diffusion actor, and θ denotes all actor-side trainable parameters, including both the observation encoder and the denoising network. The critic reuses the same latent representation z_t, but the representation is detached before it is passed to the value head. Hence the critic loss updates only the critic parameters φ and does not update the shared encoder. The detailed observation-encoder architecture is reported in Section A.1.

By Definition 3.7, a diffusion policy defines a conditional distribution over an action object. In PDPO, the action object is an H-step velocity chunk

$$x_0^t = (x_{0,1}^t, \dots, x_{0,H}^t) \in \mathbb{R}^{Hd_a}, \qquad d_a = 2, \tag{5.7}$$

where each $x_{0,j}^t \in \mathbb{R}^2$ is a velocity command. In the experiments, $H = 5$ and the control interval is 0.25 seconds, so the chunk represents a 1.25-second local plan.

Conditioned on z_t, the chunk is generated by a conditional reverse diffusion process. Starting from

$$x_K^t \sim \mathcal{N}(0, I_{Hd_a}), \tag{5.8}$$

PDPO applies Gaussian reverse transitions

$$p_\theta(x_{k-1}^t \mid x_k^t, z_t, k) = \mathcal{N}\left(x_{k-1}^t; \mu_\theta(x_k^t, z_t, k), \sigma_k^2 I_{Hd_a}\right), \qquad k = K, \dots, 1. \tag{5.9}$$

Here K denotes the denoising horizon of the current training stage.The mean μ_θ may be parameterized through the conditional noise-prediction form introduced in (3.49). The induced action-chunk distribution is

$$\pi_\theta(x_0^t \mid s_t) = \int p(x_K^t) \prod_{k=1}^{K} p_\theta(x_{k-1}^t \mid x_k^t, E_\theta(s_t), k)\, dx_1^t \cdots dx_K^t. \tag{5.10}$$

During execution, PDPO uses receding-horizon control. Only the first command of the generated chunk is applied:

$$a_t = x_{0,1}^t. \tag{5.11}$$

A new chunk is sampled at the next environment step. Thus, the diffusion model defines a distribution over short-horizon local plans, while the environment receives a standard two-dimensional velocity action.

Lemma 5.1 (Induced environment policy)**.** *Let $\pi_\theta(\cdot \mid s)$ be the action-chunk distribution induced by (5.10). The receding-horizon execution rule $a_t = x_{0,1}^t$ induces a Markov stochastic policy on the original environment MDP:*

$$\pi_\theta^{\text{exec}}(\cdot \mid s) = (\text{proj}_1)_\# \pi_\theta(\cdot \mid s), \tag{5.12}$$

where

$$\text{proj}_1(x_1, \dots, x_H) = x_1. \tag{5.13}$$

Proof. For each state s, $\pi_\theta(\cdot \mid s)$ is a probability measure on $\mathbb{R}^{Hd_a}$. The projection $\text{proj}_1 : \mathbb{R}^{Hd_a} \to \mathbb{R}^{d_a}$ is measurable, so the pushforward measure in (5.12) is a probability measure on the environment action space. Since this measure depends only on the current state s, it defines a Markov stochastic policy. □

5.2 Offline Denoising Pretraining

Before online reinforcement learning, PDPO is pretrained on expert demonstrations. This initializes the policy near feasible local behaviors before online fine-tuning. The demonstrations are obtained by rolling out ORCA in the crowd-navigation environment [Ber+11].

Let

$$\tau_{\text{exp}}^m = (s_0^m, a_0^m, s_1^m, a_1^m, \dots, s_{T_m}^m) \tag{5.14}$$

denote the m-th expert trajectory. For each valid environment time t, we construct an H-step expert action chunk

$$x_{0,\text{exp}}^{m,t} = (a_t^m, a_{t+1}^m, \dots, a_{t+H-1}^m) \in \mathbb{R}^{Hd_a}, \tag{5.15}$$

where $d_a = 2$. The resulting behavioral-cloning dataset is

$$\mathcal{D}_{\text{BC}} = \left\{ \left(s_t^m, x_{0,\text{exp}}^{m,t} \right) : m = 1, \dots, M,\ t = 0, \dots, T_m - H \right\}. \tag{5.16}$$

The policy is trained using the conditional denoising objective introduced in (3.50). For a training pair $(s_t, x_0^t) \in \mathcal{D}_{\text{BC}}$, a diffusion index and Gaussian noise are sampled according to

$$k \sim \text{Unif}\{1, \dots, K\}, \qquad \epsilon \sim \mathcal{N}(0, I_{Hd_a}). \tag{5.17}$$

Using the closed-form forward marginal from Lemma 3.2, the noisy action chunk is

$$x_k^t = \sqrt{\bar{\alpha}_k}\, x_0^t + \sqrt{1 - \bar{\alpha}_k}\, \epsilon. \tag{5.18}$$

The crowd observation is encoded as $z_t = E_\theta(s_t)$, and the denoising network predicts the injected noise from x_k^t, z_t, and k. The behavioral-cloning objective is

$$\mathcal{L}_{\text{BC}}(\theta) = \mathbb{E}_{(s_t, x_0^t) \sim \mathcal{D}_{\text{BC}},\, k, \epsilon} \left[\left\| \epsilon - \epsilon_\theta \left(x_k^t, E_\theta(s_t), k \right) \right\|_2^2 \right]. \tag{5.19}$$

Here, θ includes the parameters of both the observation encoder and the denoising network. Consequently, both components are trained jointly by minimizing $\mathcal{L}_{\text{BC}}$.

Algorithm 5 specializes the generic diffusion-model training procedure in Algorithm 4 to crowd-conditioned expert action chunks.

Algorithm 5 Offline denoising pretraining for PDPO

Require: Behavioral-cloning dataset $\mathcal{D}_{\mathrm{BC}}$, diffusion horizon K, variance schedule $\{\beta_k\}_{k=1}^{K}$, minibatch size B, number of updates N_{BC}, learning rate η

Require: Observation encoder E_θ and denoising network ϵ_θ

Compute

$$\alpha_k \leftarrow 1-\beta_k, \qquad \bar{\alpha}_k \leftarrow \prod_{i=1}^{k}\alpha_i, \qquad k=1,\dots,K$$

for $n=1,\dots,N_{\mathrm{BC}}$ **do**

  Sample a minibatch

$$\left\{(s^b, x_0^b)\right\}_{b=1}^{B} \sim \mathcal{D}_{\mathrm{BC}}$$

  for $b=1,\dots,B$ **do**

    Sample

$$k_b \sim \mathrm{Unif}\{1,\dots,K\}, \qquad \epsilon^b \sim \mathcal{N}(0, I_{Hd_a})$$

    Encode the crowd observation

$$z^b \leftarrow E_\theta(s^b)$$

    Construct the noisy action chunk

$$x_{k_b}^b \leftarrow \sqrt{\bar{\alpha}_{k_b}}\, x_0^b + \sqrt{1-\bar{\alpha}_{k_b}}\, \epsilon^b$$

  end for

  Compute the minibatch loss

$$\widehat{\mathcal{L}}_{\mathrm{BC}}(\theta) \leftarrow \frac{1}{B}\sum_{b=1}^{B}\left\|\epsilon^b - \epsilon_\theta\left(x_{k_b}^b, z^b, k_b\right)\right\|_2^2$$

  Update the actor parameters

$$\theta \leftarrow \theta - \eta\nabla_\theta\widehat{\mathcal{L}}_{\mathrm{BC}}(\theta)$$

end for

return pretrained actor parameters θ

The pretraining stage fits the conditional distribution of expert action chunks rather than a direct distribution over individual velocity commands. The resulting policy generates feasible short-horizon motion patterns conditioned on the current crowd observation and is used to initialize the subsequent online PPO stage.

5.3 The Augmented Denoising MDP

The online stage optimizes the diffusion policy through the reverse transitions that generate each action chunk. The marginal chunk density $\pi_\theta(x_0^t \mid s_t)$ requires integration over all intermediate variables, whereas every Gaussian reverse-transition density

$$p_\theta\left(x_{k-1}^t \mid x_k^t, E_\theta(s_t), k\right) \tag{5.20}$$

can be evaluated directly. Following DPPO [Ren+25], PDPO therefore represents the reverse chain as an augmented MDP.

Definition 5.2 (Augmented denoising MDP)**.** The augmented denoising MDP is

$$\overline{\mathcal{M}} = \left(\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{\mathcal{P}}_0, \bar{\mathcal{P}}, \bar{\mathcal{R}}, \bar{\gamma}\right), \tag{5.21}$$

where

$$\bar{\mathcal{S}} = \mathcal{S} \times \mathbb{R}^{Hd_a} \times \{1, \dots, K\}, \qquad \bar{\mathcal{A}} = \mathbb{R}^{Hd_a}. \tag{5.22}$$

The initial augmented state is

$$\bar{s}_{0,K} = (s_0, x_K^0, K), \qquad s_0 \sim \mathcal{P}_0, \qquad x_K^0 \sim \mathcal{N}(0, I_{Hd_a}), \tag{5.23}$$

which defines $\bar{\mathcal{P}}_0$.

At environment time t and denoising level k, the augmented state and action are

$$\bar{s}_{t,k} = (s_t, x_k^t, k), \qquad \bar{a}_{t,k} = x_{k-1}^t, \tag{5.24}$$

and the augmented policy is

$$\bar{\pi}_\theta\left(\bar{a}_{t,k} \mid \bar{s}_{t,k}\right) = p_\theta\left(x_{k-1}^t \mid x_k^t, E_\theta(s_t), k\right). \tag{5.25}$$

For $k > 1$, the transition is deterministic after sampling the augmented action:

$$\bar{s}_{t,k-1} = (s_t, x_{k-1}^t, k-1), \tag{5.26}$$

and the reward is zero. At $k = 1$, the generated chunk x_0^t yields the environment action $a_t = \mathrm{proj}_1(x_0^t)$. The next augmented state is

$$\bar{s}_{t+1,K} = (s_{t+1}, x_K^{t+1}, K), \tag{5.27}$$

where

$$s_{t+1} \sim \mathcal{P}\left(\cdot \mid s_t, \mathrm{proj}_1(x_0^t)\right), \qquad x_K^{t+1} \sim \mathcal{N}(0, I_{Hd_a}). \tag{5.28}$$

These rules define the transition kernel $\bar{\mathcal{P}}$.

The augmented reward and transition discount are

$$\bar{\mathcal{R}}(\bar{s}_{t,k}, \bar{a}_{t,k}) = \begin{cases} 0, & k > 1, \\ \mathcal{R}\left(s_t, \mathrm{proj}_1(x_0^t)\right), & k = 1, \end{cases} \tag{5.29}$$

and

$$\bar{\gamma}(\bar{s}_{t,k}, \bar{a}_{t,k}) = \begin{cases} 1, & k > 1, \\ \gamma, & k = 1. \end{cases} \tag{5.30}$$

Thus, discounting is applied when the process advances to the next environment time, not between denoising transitions within the same environment step.

The augmented MDP introduces no additional task reward inside the sampler. Its intermediate rewards are zero, and its terminal reward is exactly the reward of the outer crowd-navigation MDP.

For each environment step t, let

$$\xi_t = (x_K^t, x_{K-1}^t, \dots, x_0^t) \tag{5.31}$$

denote the sampled reverse trajectory. Conditioned on s_t, its density is

$$p_\theta(\xi_t \mid s_t) = p(x_K^t) \prod_{k=1}^{K} p_\theta\left(x_{k-1}^t \mid x_k^t, E_\theta(s_t), k\right). \tag{5.32}$$

For a finite outer horizon T, define

$$\bar{\tau}_{0:T} = (s_0, \xi_0, r_0, \ldots, s_{T-1}, \xi_{T-1}, r_{T-1}, s_T). \tag{5.33}$$

Its density factorizes as

$$p_\theta(\bar{\tau}_{0:T}) = \mathcal{P}_0(s_0) \prod_{t=0}^{T-1} \left[p_\theta(\xi_t \mid s_t)\, \mathcal{P}\left(s_{t+1} \mid s_t, \mathrm{proj}_1(x_0^t)\right)\right]. \tag{5.34}$$

This is the analogue of (2.7) for the coupled outer–inner process.

Proposition 5.3 (Score decomposition)**.** *For the finite augmented trajectory in* (5.34)*,*

$$\nabla_\theta \log p_\theta(\bar{\tau}_{0:T}) = \sum_{t=0}^{T-1} \sum_{k=1}^{K} \nabla_\theta \log p_\theta \left(x_{k-1}^t \mid x_k^t, E_\theta(s_t), k\right). \tag{5.35}$$

Proof. Taking logarithms in (5.34) gives

$$\begin{aligned} \log p_\theta(\bar{\tau}_{0:T}) = \log \mathcal{P}_0(s_0) &+ \sum_{t=0}^{T-1} \log p_\theta(\xi_t \mid s_t) \\ &+ \sum_{t=0}^{T-1} \log \mathcal{P}\left(s_{t+1} \mid s_t, \mathrm{proj}_1(x_0^t)\right). \end{aligned} \tag{5.36}$$

The initial-state distribution and environment transition kernel do not depend explicitly on θ once the sampled denoising trajectory is fixed. By (5.32),

$$\log p_\theta(\xi_t \mid s_t) = \log p(x_K^t) + \sum_{k=1}^{K} \log p_\theta \left(x_{k-1}^t \mid x_k^t, E_\theta(s_t), k\right), \tag{5.37}$$

and the initial-noise density is independent of θ. Differentiating therefore gives (5.35). □

Equation (5.35) allows the likelihood-ratio score to be evaluated at each denoising transition without evaluating the marginal chunk density.

Corollary 5.4 (Finite-horizon denoising-level policy gradient)**.** *Define*

$$J_T(\theta) = \mathbb{E}_{\bar{\tau}_{0:T} \sim p_\theta} \left[\sum_{t=0}^{T-1} \gamma^t \mathcal{R}\left(s_t, \mathrm{proj}_1(x_0^t)\right)\right]. \tag{5.38}$$

Under the regularity assumptions of Assumption 2.19,

$$\nabla_\theta J_T(\theta) = \mathbb{E}_{\bar{\tau}_{0:T} \sim p_\theta} \left[\sum_{t=0}^{T-1} \gamma^t \sum_{k=1}^{K} \nabla_\theta \log p_\theta \left(x_{k-1}^t \mid x_k^t, E_\theta(s_t), k\right) G_t^{(T)} \right], \tag{5.39}$$

where

$$G_t^{(T)} = \sum_{\ell=0}^{T-1-t} \gamma^\ell \mathcal{R}\left(s_{t+\ell}, \mathrm{proj}_1(x_0^{t+\ell})\right). \tag{5.40}$$

Proof. Apply the likelihood-ratio identity in Lemma 2.21 to (5.38) and use Proposition 5.3. By causality, the reverse trajectory sampled at environment time t can affect only the reward at time t and later rewards. The corresponding return-to-go is therefore $G_t^{(T)}$. □

Under the bounded-reward and regularity assumptions used in Theorem 2.25, taking $T \to \infty$ gives

$$J(\theta) = \mathbb{E}_{\bar{\tau}\sim p_\theta}\left[\sum_{t\geq 0}\gamma^t \mathcal{R}\left(s_t, \mathrm{proj}_1(x_0^t)\right)\right], \tag{5.41}$$

and

$$\nabla_\theta J(\theta) = \mathbb{E}_{\bar{\tau}\sim p_\theta}\left[\sum_{t\geq 0}\gamma^t \sum_{k=1}^{K} \nabla_\theta \log p_\theta\left(x_{k-1}^t \mid x_k^t, E_\theta(s_t), k\right) G_t\right], \tag{5.42}$$

where

$$G_t = \sum_{\ell=0}^{\infty}\gamma^\ell \mathcal{R}\left(s_{t+\ell}, \mathrm{proj}_1(x_0^{t+\ell})\right). \tag{5.43}$$

In implementation, G_t is replaced by an environment-level advantage estimate, which is assigned to the denoising transitions that generated x_0^t.

Algorithm 6 gives the rollout procedure used to sample the coupled process. In addition to ordinary environment transitions, the buffer stores every denoising transition and its log density under the rollout policy.

Algorithm 6 Collecting an augmented PDPO rollout

Require: Environment $\mathcal{M}_{\text{env}}$, rollout actor θ_{old}, diffusion horizon K, rollout length M

Initialize environment buffer $\mathcal{B}_{\text{env}} \leftarrow \emptyset$

Initialize denoising buffer $\mathcal{B}_{\text{den}} \leftarrow \emptyset$

Sample $s_0 \sim \mathcal{P}_0$

for $t = 0, \dots, M-1$ **do**

  Encode the crowd observation

$$z_t \leftarrow E_{\theta_{\text{old}}}(s_t)$$

  Sample initial diffusion noise

$$x_K^t \sim \mathcal{N}(0, I_{Hd_a})$$

  for $k = K, K-1, \dots, 1$ **do**

    Sample the next denoising variable

$$x_{k-1}^t \sim p_{\theta_{\text{old}}}\left(\cdot \mid x_k^t, z_t, k\right)$$

    Evaluate the rollout-policy log density

$$\ell_{t,k}^{\text{old}} \leftarrow \log p_{\theta_{\text{old}}}\left(x_{k-1}^t \mid x_k^t, z_t, k\right)$$

    Store

$$\left(s_t, x_k^t, x_{k-1}^t, k, \ell_{t,k}^{\text{old}}\right)$$

  in $\mathcal{B}_{\text{den}}$

  end for

  Execute the first command of the generated chunk

$$a_t \leftarrow \text{proj}_1(x_0^t)$$

  Apply a_t and observe r_t, s_{t+1}, and terminal indicator d_t

  Store

$$(s_t, r_t, s_{t+1}, d_t)$$

  in $\mathcal{B}_{\text{env}}$

  if $d_t = 1$ **then**

    Reset the environment and sample $s_{t+1} \sim \mathcal{P}_0$

  end if

end for

return $\mathcal{B}_{\text{env}}$ and $\mathcal{B}_{\text{den}}$

Remark 5.5 (Role of the two time scales). The index t denotes environment time, whereas k denotes the position within the reverse denoising chain. Rewards, value estimates, and environment discounting are defined at the outer time scale. Policy likelihoods are evaluated at the inner time scale. The online update described next combines these two quantities.

5.4 Online PPO Fine-Tuning

After denoising pretraining, PDPO fine-tunes the actor through interaction with the crowd-navigation environment. The update follows PPO [Sch+17] and the transition-level diffusion-

policy optimization used in DPPO [Ren+25]. The critic remains an environment-level value function $V_\varphi(s_t)$. The actor, however, is updated through the likelihoods of the reverse denoising transitions that generated the executed action chunk.

For a rollout stored in $\mathcal{B}_{\text{env}}$, define the temporal-difference residual

$$\delta_t^\varphi = r_t + \gamma(1 - d_t)V_\varphi(s_{t+1}) - V_\varphi(s_t), \tag{5.44}$$

where $d_t \in \{0, 1\}$ indicates whether the episode terminates after time t. The environment-level advantage is estimated using GAE:

$$\widehat{A}_t^{\text{env}} = \sum_{\ell=0}^{M-1-t} (\gamma\lambda)^\ell \left(\prod_{j=0}^{\ell-1} (1 - d_{t+j}) \right) \delta_{t+\ell}^\varphi. \tag{5.45}$$

Equivalently, it may be computed recursively as

$$\widehat{A}_t^{\text{env}} = \delta_t^\varphi + \gamma\lambda(1 - d_t)\widehat{A}_{t+1}^{\text{env}}. \tag{5.46}$$

The corresponding critic target is

$$\widehat{G}_t = \widehat{A}_t^{\text{env}} + V_\varphi(s_t). \tag{5.47}$$

The actor update uses the denoising transitions stored in $\mathcal{B}_{\text{den}}$. Each entry corresponds to an augmented state–action pair

$$\bar{s}_{t,k} = (s_t, x_k^t, k), \qquad \bar{a}_{t,k} = x_{k-1}^t. \tag{5.48}$$

For data collected under the rollout actor θ_{old}, the transition-level likelihood ratio is

$$\begin{aligned} r_{t,k}(\theta) &= \frac{\bar{\pi}_\theta\left(\bar{a}_{t,k} \mid \bar{s}_{t,k}\right)}{\bar{\pi}_{\theta_{\text{old}}}\left(\bar{a}_{t,k} \mid \bar{s}_{t,k}\right)} \\ &= \frac{p_\theta\left(x_{k-1}^t \mid x_k^t, E_\theta(s_t), k\right)}{p_{\theta_{\text{old}}}\left(x_{k-1}^t \mid x_k^t, E_{\theta_{\text{old}}}(s_t), k\right)}. \end{aligned} \tag{5.49}$$

During rollout collection, the old-policy log density $\ell_{t,k}^{\text{old}}$ is stored for each denoising transition. Therefore, the ratio used during PPO updates can be evaluated as

$$r_{t,k}(\theta) = \exp\left[\ell_{t,k}(\theta) - \ell_{t,k}^{\text{old}}\right], \qquad \ell_{t,k}(\theta) = \log p_\theta\left(x_{k-1}^t \mid x_k^t, E_\theta(s_t), k\right). \tag{5.50}$$

The environment-level advantage is assigned to all denoising transitions that generated the corresponding action chunk. This follows from the reward structure in Definition 5.2: intermediate denoising transitions have zero reward, while the terminal denoising transition inherits the environment reward. Following DPPO, we allow a denoising weighting factor $\gamma_{\text{denoise}} \in (0, 1]$ to scale the learning signal across denoising levels:

$$\widehat{A}_{t,k} = \gamma_{\text{denoise}}^{k-1} \widehat{A}_t^{\text{env}}. \tag{5.51}$$

This factor is not the environment discount. The environment discount γ determines the return across outer time steps, whereas γ_{denoise} only changes how strongly the same environment-level advantage is assigned to different reverse denoising transitions. Since the reverse chain is sampled from K to 1, larger values of k correspond to earlier and noisier transitions. Choosing $\gamma_{\text{denoise}} < 1$ therefore gives smaller weights to these earlier denoising steps.

The clipped actor objective is

$$\mathcal{L}_{\mathrm{PPO}}^{\mathrm{den}}(\theta) = \mathbb{E}_{t,k}\Big[\min\Big\{r_{t,k}(\theta)\widehat{A}_{t,k}, \mathrm{clip}\left(r_{t,k}(\theta), 1-\varepsilon_{\mathrm{PPO}}, 1+\varepsilon_{\mathrm{PPO}}\right)\widehat{A}_{t,k}\Big\}\Big]. \tag{5.52}$$

This is the clipped PPO objective from Definition 2.34, applied to the reverse-transition policy rather than directly to the marginal environment-action policy. The critic is trained with the environment-level squared loss

$$\mathcal{L}_{\mathrm{value}}(\varphi) = \mathbb{E}_t\left[\left(V_\varphi(s_t) - \widehat{G}_t\right)^2\right]. \tag{5.53}$$

In the implementation, the actor is updated by maximizing $\mathcal{L}_{\mathrm{PPO}}^{\mathrm{den}}$, while the critic is updated by minimizing $\mathcal{L}_{\mathrm{value}}$.

Remark 5.6 (Trajectory and transition ratios). The likelihood ratio of a complete denoising trajectory factorizes as

$$\frac{p_\theta(\xi_t \mid s_t)}{p_{\theta_{\mathrm{old}}}(\xi_t \mid s_t)} = \prod_{k=1}^{K} r_{t,k}(\theta). \tag{5.54}$$

PDPO does not clip this product directly. Instead, it applies PPO clipping to the individual transition ratios, following DPPO [Ren+25]. This avoids forming a product whose variability may increase with the number of denoising steps. The resulting update should be understood as a transition-level PPO surrogate for optimizing the sampled denoising trajectory distribution, rather than as an exact PPO update on the intractable marginal chunk density.

Algorithm 7 summarizes the online fine-tuning procedure. The rollout stage stores both environment transitions and denoising transitions. The return and value targets are computed at the environment level, while the actor objective is evaluated on minibatches of denoising transitions.

The two training stages serve different roles. Denoising pretraining fits the expert action-chunk distribution and initializes feasible local behavior. Online PPO then adjusts the same crowd-conditioned denoising policy using the environment reward. The next chapter evaluates this policy in the CrowdNav++ simulator and studies the effect of action chunks through ablations and qualitative visualizations.

Algorithm 7 Online PPO fine-tuning for PDPO

Require: Pretrained actor θ, critic φ, rollout length M, diffusion horizon K
Require: Discounts γ, λ, denoising weight γ_{denoise}, PPO clip ε_{PPO}
Require: Learning rates $\eta_\theta, \eta_\varphi$, PPO epochs N_{PPO}

for each training iteration **do**
  Set rollout actor $\theta_{\text{old}} \leftarrow \theta$
  Collect $\mathcal{B}_{\text{env}}$ and $\mathcal{B}_{\text{den}}$ using Algorithm 6
  Compute $\widehat{A}_t^{\text{env}}$ and $\widehat{G}_t$ from $\mathcal{B}_{\text{env}}$ using (5.44)–(5.47)
  Assign denoising advantages

$$\widehat{A}_{t,k} \leftarrow \gamma_{\text{denoise}}^{k-1} \widehat{A}_t^{\text{env}}, \qquad (t,k) \in \mathcal{B}_{\text{den}}$$

  for $e = 1, \ldots, N_{\text{PPO}}$ **do**
    for each minibatch $\mathcal{M}_{\text{env}} \subset \mathcal{B}_{\text{env}}$, $\mathcal{M}_{\text{den}} \subset \mathcal{B}_{\text{den}}$ **do**
      Recompute current log densities $\ell_{t,k}(\theta)$ for $(t,k) \in \mathcal{M}_{\text{den}}$
      Compute ratios $r_{t,k}(\theta)$ using (5.50)
      Compute actor objective $\widehat{\mathcal{L}}_{\text{PPO}}^{\text{den}}(\theta)$ using (5.52)
      Compute critic loss $\widehat{\mathcal{L}}_{\text{value}}(\varphi)$ using (5.53)
      Update actor:

$$\theta \leftarrow \theta + \eta_\theta \nabla_\theta \widehat{\mathcal{L}}_{\text{PPO}}^{\text{den}}(\theta)$$

      Update critic:

$$\varphi \leftarrow \varphi - \eta_\varphi \nabla_\varphi \widehat{\mathcal{L}}_{\text{value}}(\varphi)$$

    end for
  end for
end for
return fine-tuned actor θ and critic φ

Chapter 6

Experiments

This chapter evaluates Planning Diffusion Policy Optimization (PDPO) in dense robot crowd-navigation tasks. The experiments are designed to answer three questions. First, does the proposed diffusion-policy formulation perform competitively against classical and learning-based crowd-navigation baselines? Second, which parts of the training pipeline are needed to obtain a stable and effective policy? Third, does representing the policy output as a short-horizon action chunk provide an advantage over a diffusion policy that generates only a single action?

All experiments are conducted in the CrowdNav++ simulation environment [Liu+23]. We evaluate policies under both the original CrowdNav++ protocol and a bounded variant in which workspace boundary violations are treated as collisions. The original protocol is retained because it permits direct comparison with previously reported results. The bounded protocol is introduced to test whether a policy succeeds while remaining inside the intended navigation workspace.

The chapter is organized as follows. We first describe the simulator, the two evaluation protocols, the baselines, and the training procedure. We then document the empirical development of the final offline-to-online pipeline, including the role of behavioral-cloning initialization and the transition from single-action diffusion policies to short action chunks. The main quantitative evaluation compares PDPO with rule-based and learning-based baselines. We then analyze boundary effects, isolate the contribution of action chunks through an ablation study, and visualize generated short-horizon plans. Additional implementation details, network architectures, hyperparameters, model sizes, and checkpoint-selection procedures are reported in Chapter A.

6.1 Experimental Setup

6.1.1 Simulation Environment

The experiments use the CrowdNav++ simulator [Liu+23], a benchmark for robot navigation in dense and dynamic human crowds. This environment is appropriate for the present study because it combines continuous robot control, variable-size crowd observations, and nontrivial local interaction patterns. These properties make it a suitable testbed for comparing single-step action policies with short-horizon action-chunk policies.

The environment consists of a two-dimensional 12×12 m^2 workspace containing one robot and 20 human agents. The robot must reach its goal while avoiding collisions with humans and maintaining socially acceptable navigation behavior. Figure 6.1 illustrates the basic environment layout.

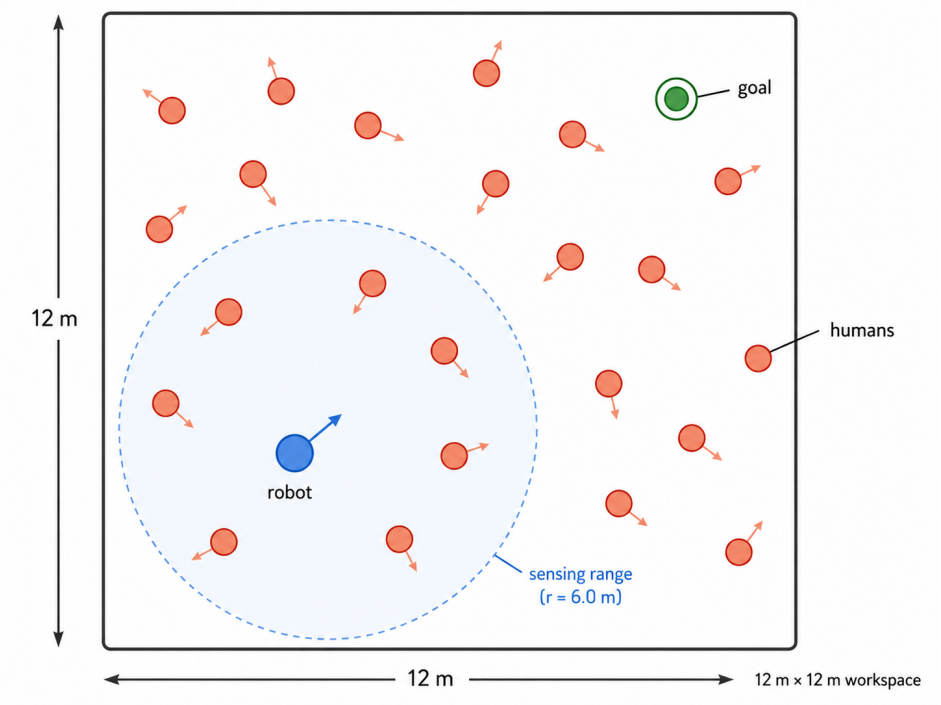


Figure 6.1: Illustration of the CrowdNav++ environment. The robot navigates in a 12×12 m^2 workspace populated by 20 human agents. The dashed circle indicates the robot's sensing range ($r = 6.0$ m), and the green marker denotes the robot's goal.

At the beginning of each episode, the robot and all humans are initialized with random positions and goals. Human agents use ORCA for collision avoidance among themselves, but they do not observe or react to the robot. The robot therefore navigates under one-sided interaction and cannot rely on cooperative human responses.

The robot is modeled as a circular agent with radius 0.2 m, maximum speed 1.0 m/s, and sensing range 6.0 m. Human radii are sampled uniformly from $[0.3, 0.5]$ m, and their maximum speeds are sampled from $[0.5, 1.5]$ m/s. At each decision step, the robot observes nearby humans within its sensing range together with their motion states, while the humans continue to move according to their ORCA-based policies.

To generate dynamic crowd flows, human agents are assigned new goal locations after reaching their current goals. In addition, at each second, each human independently changes its goal with probability 0.5. Each episode terminates when the robot reaches its goal, collides with a human, or exceeds the maximum episode length of 50 seconds. Episodes that reach the time limit are treated as timeouts.

6.1.2 Original and Bounded Evaluation Protocols

We consider two variants of the simulator. The first is the original CrowdNav++ benchmark protocol, which does not treat workspace boundary violations as collisions. This setting is useful because it permits direct comparison with results reported in prior work. However, it does not fully distinguish trajectories that remain inside the nominal workspace from trajectories that temporarily leave the workspace and later reach the goal.

The second setting is a bounded variant, referred to as *Bounded CrowdNav*. The crowd-generation process, robot dynamics, human policies, and evaluation metrics remain unchanged, while a trajectory that crosses the workspace boundary is terminated and

counted as a collision. This variant tests a stricter form of navigation success: the robot must reach the goal while remaining inside the intended domain.

The bounded protocol was introduced after trajectory inspection showed that some successful episodes in the original benchmark left the nominal workspace. The boundary diagnosis in Section 6.3.2 quantifies this effect and motivates the use of Bounded CrowdNav as the main stricter evaluation setting.

6.1.3 Baselines and Ablations

The baselines are chosen to separate three directions in crowd-navigation research. ORCA represents classical rule-based collision avoidance in velocity space. CrowdNav++ represents the learning-based state-representation direction, where prediction, attention, interaction modeling, and temporal recurrence are used to build a stronger encoder for model-free reinforcement learning. PDPO instead focuses on the action-generation side by replacing a single-step action policy with a diffusion policy over short-horizon velocity chunks.

We compare PDPO with two representative crowd-navigation baselines:

- **ORCA**: a classical rule-based collision-avoidance method [Ber+11]. ORCA is included as a non-learning reference point. It provides a useful comparison because it performs local collision avoidance through geometric velocity constraints, without learned state representation or learned action generation.
- **CrowdNav++ (Const. Vel.)**: a learning-based crowd-navigation baseline using a constant-velocity human trajectory predictor [Liu+23]. CrowdNav++ is included as the main learning-based reference point because it uses the same simulator and PPO training framework, but emphasizes prediction-aware state representation rather than diffusion-based action generation.

We additionally evaluate two variants of PDPO:

- **PDPO w/o action chunk**: a diffusion-policy variant that generates a single action at each control step. This ablation keeps the diffusion-policy parameterization and online PPO training procedure, but changes the generated action object from a short sequence to one instantaneous velocity command. It tests whether the gain comes from action-chunk generation rather than from using a diffusion actor alone.
- **PDPO from scratch**: the full action-chunk policy trained online from random initialization without behavioral-cloning pretraining. This variant tests whether offline denoising pretraining is needed for sample-efficient online reinforcement learning.

Unless otherwise stated, all learning-based methods use the same constant-velocity predictor to forecast human motion over the next five timesteps. This keeps the prediction component fixed and focuses the comparison on the policy representation and training procedure.

6.1.4 Training and Evaluation Protocol

The final PDPO policy and the single-action ablation follow an offline-to-online training procedure. This design is used because direct online reinforcement learning with a diffusion actor is sample-intensive in the dense crowd setting, where early random policies frequently collide or time out. Behavioral cloning provides an initial policy that already generates feasible local motions, after which PPO can adapt the policy to the environment reward.

The diffusion policy is first pretrained by behavioral cloning on approximately 3,000 demonstration episodes. At each timestep, the observation and executed velocity command are recorded. For the full PDPO policy, expert action chunks are constructed from the current and subsequent velocity commands. The single-action ablation uses only the current command. Behavioral-cloning pretraining runs for 200 optimization iterations.

The source of the demonstrations depends on the evaluation protocol. In the original CrowdNav++ environment, ORCA generates the behavioral-cloning trajectories. This choice matches the classical baseline and provides a rule-based source of collision-avoidance behavior. In Bounded CrowdNav, ORCA does not account for the enforced workspace boundary and consequently produces frequent boundary collisions. We therefore use a trained CrowdNav++ policy to generate demonstrations within the bounded workspace. This keeps the offline data consistent with the constraints of the bounded evaluation protocol. These trajectories form the offline pretraining dataset, after which the PDPO policies are fine-tuned through online interaction with the environment.

After pretraining, the policies are optimized online with PPO for approximately 12 million environment timesteps. The full PDPO policy uses an action chunk length of $H = 5$. Since the simulator has a control interval of 0.25 s, each chunk represents a 1.25 s short-horizon plan. Only the first action of the sampled chunk is applied, and a new chunk is generated at the next timestep. The single-action ablation uses $H = 1$, while the from-scratch variant uses $H = 5$ and omits the behavioral-cloning stage.

For the original benchmark, the ORCA and CrowdNav++ results are taken from Liu et al. [Liu+23], since the environment and evaluation protocol are unchanged. For Bounded CrowdNav, all learning-based methods are trained and evaluated under the bounded protocol.

We evaluate navigation performance using the following metrics:

- **Success rate (SR)**: the percentage of episodes in which the robot reaches its goal without collision or timeout.
- **Navigation time (NT)**: the time required to reach the goal in successful episodes.
- **Path length (PL)**: the total distance traveled by the robot.
- **Intrusion time ratio (ITR)**: for each episode, the fraction of timesteps at which the robot intrudes into any human's ground-truth future personal zones over the next five steps, averaged over all test episodes.
- **Social distance during intrusions (SD)**: the average distance between the robot and its closest human at timesteps when an intrusion occurs.

Success rate measures task completion, while navigation time and path length measure efficiency. ITR and SD provide complementary information about crowd-aware safety. All intrusions are evaluated using the ground-truth future human positions. Each final policy is evaluated on 500 unseen test seeds. For learning-based methods in Bounded CrowdNav, results are reported over three random training seeds. Details of the observation encoder, diffusion network, value function, optimizer settings, and checkpoint selection are given in Chapter A.

6.2 Development of the Training Pipeline

This section documents the empirical choices that led to the final training pipeline. It explains why the final method uses offline denoising pretraining, why the demonstration

source differs between the original and bounded protocols, and why the final actor generates short action chunks rather than single actions.

The first design question is whether the full action-chunk diffusion policy can be trained directly from online interaction. To test this, we trained the action-chunk policy from random initialization on Bounded CrowdNav. As shown in Figure 6.2, the policy acquired useful navigation behavior slowly and remained at a low success level for much of the available interaction budget. This result motivated the use of behavioral-cloning initialization before online PPO fine-tuning.

Behavioral cloning provides a useful but limited initialization. Before online fine-tuning, the pretrained policy reaches a success rate of approximately 45% in the original environment and approximately 40% in Bounded CrowdNav. These policies already exhibit basic goal-directed and collision-avoidance behavior, providing a stronger starting point for online reinforcement learning. The change from ORCA demonstrations in the original environment to CrowdNav++ demonstrations in the bounded environment also adapts the offline data to the corresponding workspace constraints.

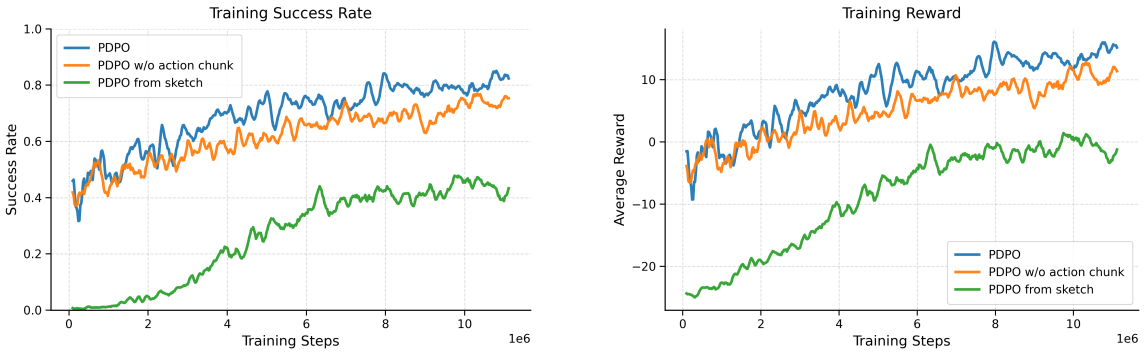


Figure 6.2: Training behavior on Bounded CrowdNav for behaviorally pretrained PDPO with five-step action chunks, the corresponding action-chunk policy trained from scratch, and the behaviorally pretrained single-action diffusion-policy ablation. Left: training success rate. Right: average training reward. Behavioral-cloning initialization accelerates learning relative to direct training from random initialization. Both pretrained variants undergo a short transition at the beginning of online fine-tuning and subsequently improve through continued environment interaction.

The pretrained policies exhibit a short performance decrease when online PPO fine-tuning begins, followed by rapid recovery and continued improvement. This transient behavior coincides with the shift from supervised imitation on a fixed demonstration distribution to reinforcement learning under the state distribution induced by the evolving policy. The objective also changes from reproducing teacher actions to maximizing environment return.

The second design question is whether diffusion should be applied to instantaneous actions or to short-horizon action sequences. The single-action diffusion policy provides an intermediate design: it tests whether a diffusion actor can be optimized successfully without changing the policy output into a sequence. Its training curve and final evaluation show that a single-action diffusion actor can learn a competent policy, with performance close to CrowdNav++. The larger improvement appears when the diffusion policy is defined over short-horizon action sequences.

This observation matches the structure of crowd-navigation decisions. In dense crowds, multimodality often concerns alternative local maneuvers—such as passing on different sides of a pedestrian, slowing before entering a gap, or making a short detour—rather than isolated instantaneous velocities. Action chunking moves the policy distribution from

single-step action space to short-horizon trajectory space, where each sampled mode can encode a coherent local maneuver.

These observations led to the final pipeline used in the main experiments: behavioral-cloning initialization with a demonstrator adapted to the evaluation protocol, followed by online PPO fine-tuning of a diffusion policy over five-step action chunks. The following sections evaluate the resulting policies on held-out test episodes.

6.3 Quantitative Evaluation

6.3.1 Original CrowdNav++ Benchmark

We first evaluate PDPO under the original CrowdNav++ benchmark protocol. This experiment serves a compatibility purpose: it allows PDPO to be compared with the previously reported ORCA and CrowdNav++ results under the same protocol. For this reason, the original benchmark remains informative even though Section 6.3.2 later shows that its handling of workspace boundaries affects the interpretation of success rates.

Table 6.1: Navigation performance on the original CrowdNav++ benchmark without explicit boundary constraints. ORCA and CrowdNav++ results are reported from Liu et al. [Liu+23].

Method	**SR↑**	**NT↓**	**PL↓**	**ITR↓**	**SD↑**
ORCA	69.0	14.77	17.67	19.61	0.38
CrowdNav++ (Const. Vel.)	87.0	14.03	20.14	7.00	0.42
PDPO w/o action chunk	86.2	14.59	19.20	8.04	0.40
PDPO (Ours)	**90.6**	14.50	20.25	7.22	0.41

As shown in Table 6.1, PDPO achieves a success rate of 90.6%, improving over the reported CrowdNav++ result by 3.6 percentage points. The single-action diffusion-policy variant reaches 86.2%, close to the CrowdNav++ result. Adding action chunks increases the success rate from 86.2% to 90.6%.

The full PDPO policy achieves the highest success rate while remaining competitive on the remaining navigation metrics. The difference between the single-action and action-chunk variants also indicates that the structure of the generated action object contributes to the final performance. The next subsection examines why the original success rate should be interpreted together with boundary behavior.

6.3.2 Boundary Diagnosis

The purpose of this diagnostic experiment is to check whether success in the original benchmark always corresponds to successful navigation inside the intended workspace. Inspection of successful rollouts revealed that some trajectories left the nominal 12×12 m^2 workspace and later reached the goal. Since the original protocol does not classify boundary violations as failures, these trajectories remain part of the reported success rate. We quantify this effect by measuring the fraction of successful episodes that cross the workspace boundary.

Table 6.2 shows that out-of-bound trajectories occur across all evaluated methods. Among successful episodes, 31.8% of ORCA rollouts leave the workspace. The corresponding fractions are 16.1% for CrowdNav++, 18.7% for PDPO without action chunks, and 19.8% for the full PDPO policy. The success rate in the original protocol therefore combines

Table 6.2: Fraction of successful trajectories in the original benchmark that leave the nominal workspace while still being counted as successes.

Method	Out-of-bound successful trajectories
ORCA	31.8%
CrowdNav++ (Const. Vel.)	16.1%
PDPO w/o action chunk	18.7%
PDPO (Ours)	19.8%

trajectories that remain inside the intended domain with trajectories that leave the nominal workspace.

This observation motivated the Bounded CrowdNav protocol introduced in Section 6.1.2. The bounded evaluation requires retraining because the wall constraint changes both the failures encountered during policy learning and the quality of the available demonstration trajectories.

6.3.3 Bounded CrowdNav

Bounded CrowdNav is used as the stricter evaluation setting. The motivation is not to change the crowd-generation mechanism or the robot dynamics, but to ensure that a successful episode corresponds to reaching the goal inside the valid navigation region. Under this protocol, all learning-based policies are trained and evaluated with boundary violations treated as collisions.

Table 6.3: Navigation performance on Bounded CrowdNav. All learning-based methods are trained and evaluated under the same bounded protocol. SR, NT, PL, and ITR are reported as mean and standard deviation over three random training seeds; SD is reported as the aggregate mean. ORCA is deterministic and is reported without a standard deviation.

Method	SR↑	NT↓	PL↓	ITR↓	SD↑
ORCA	47.0	21.26	17.14	1.35	0.50
CrowdNav++ (Const. Vel.)	74.5 ± 2.1	13.35 ± 1.13	18.42 ± 0.65	8.53 ± 0.30	0.42
PDPO w/o action chunk	73.7 ± 0.7	13.05 ± 0.46	20.93 ± 0.50	7.21 ± 0.74	0.40
PDPO (Ours)	$\mathbf{84.7 \pm 1.4}$	$\mathbf{10.92 \pm 0.04}$	20.58 ± 0.19	$\mathbf{6.81 \pm 0.04}$	0.41

Table 6.3 shows a clearer separation between the methods. PDPO reaches a success rate of 84.7%, outperforming CrowdNav++ by 10.2 percentage points and the single-action diffusion-policy variant by 11.0 percentage points. It also achieves the lowest navigation time and intrusion rate among the learning-based methods.

PDPO travels a longer path than CrowdNav++, with 20.58 m compared with 18.42 m. At the same time, it achieves a shorter navigation time and a lower intrusion rate. The additional distance is therefore associated with more lateral or avoidance motion while maintaining efficient progress toward the goal.

The bounded results also reflect the limitations of ORCA in the modified environment. Its success rate falls to 47.0%, and trajectory inspection shows frequent boundary collisions. This observation motivated the use of a trained CrowdNav++ policy to generate the bounded demonstration dataset. In other words, the demonstration policy is chosen to match the constraints of the evaluation protocol.

6.4 Ablation Study: Role of Action Chunks

The main methodological claim of PDPO is not only that diffusion policies are expressive, but that diffusion over short action chunks is useful for crowd navigation. The single-action ablation tests this claim by keeping the diffusion parameterization, behavioral-cloning initialization, and online PPO fine-tuning procedure, while changing the generated action object from a sequence to one velocity command.

The training curves in Figure 6.2 describe how the single-action and action-chunk policies evolve during optimization. We now use the final held-out evaluations to measure the effect of action chunking. Both variants use a diffusion-policy parameterization, behavioral-cloning initialization, and online PPO fine-tuning. Their main difference is the object generated by the policy: one velocity action for the ablation and a five-step local plan for PDPO.

Table 6.4: Effect of action chunking on success rate. The improvement is measured relative to the single-action diffusion-policy ablation.

Setting	**PDPO w/o action chunk**	**PDPO**	**Improvement**
Original CrowdNav++	86.2	90.6	+4.4
Bounded CrowdNav	73.7	84.7	+11.0

As summarized in Table 6.4, action chunking improves success rate in both settings. The increase is 4.4 percentage points in the original benchmark and 11.0 percentage points in Bounded CrowdNav. The larger difference in the bounded setting indicates a stronger benefit from short-horizon action generation when the robot must remain inside the valid workspace.

For a single-action policy, multimodality is represented over instantaneous velocity commands. Action chunking extends this distribution to short motion sequences, allowing alternative local maneuvers to be represented directly. Passing to the left, passing to the right, slowing down to yield, and entering a temporary gap are more naturally distinguished as action sequences than as isolated first actions. The receding-horizon execution scheme then preserves closed-loop responsiveness by replanning after every applied action.

The ablation is consistent with the view that the performance gain arises from combining a multimodal diffusion policy with a short-horizon sequence representation. Each sampled mode describes a coherent local plan, while the repeated replanning step adapts that plan to the evolving crowd configuration.

6.5 Qualitative Analysis of Action Chunks

The quantitative ablation measures whether action chunks improve navigation performance. The qualitative analysis examines what the learned action chunks look like. In particular, it asks whether the generated chunks form temporally coherent short-horizon motions and whether multiple samples under the same observation can represent different feasible local plans.

In all visualizations, each snapshot is centered on the robot. The dashed circle indicates the robot's observation range. Humans within this range are shown as blue dots together with their predicted linear trajectories. At each decision step, the policy generates a five-step action chunk, with each action corresponding to 0.25 s. Each displayed chunk therefore covers the next 1.25 s. Only its first action is executed before the policy replans.

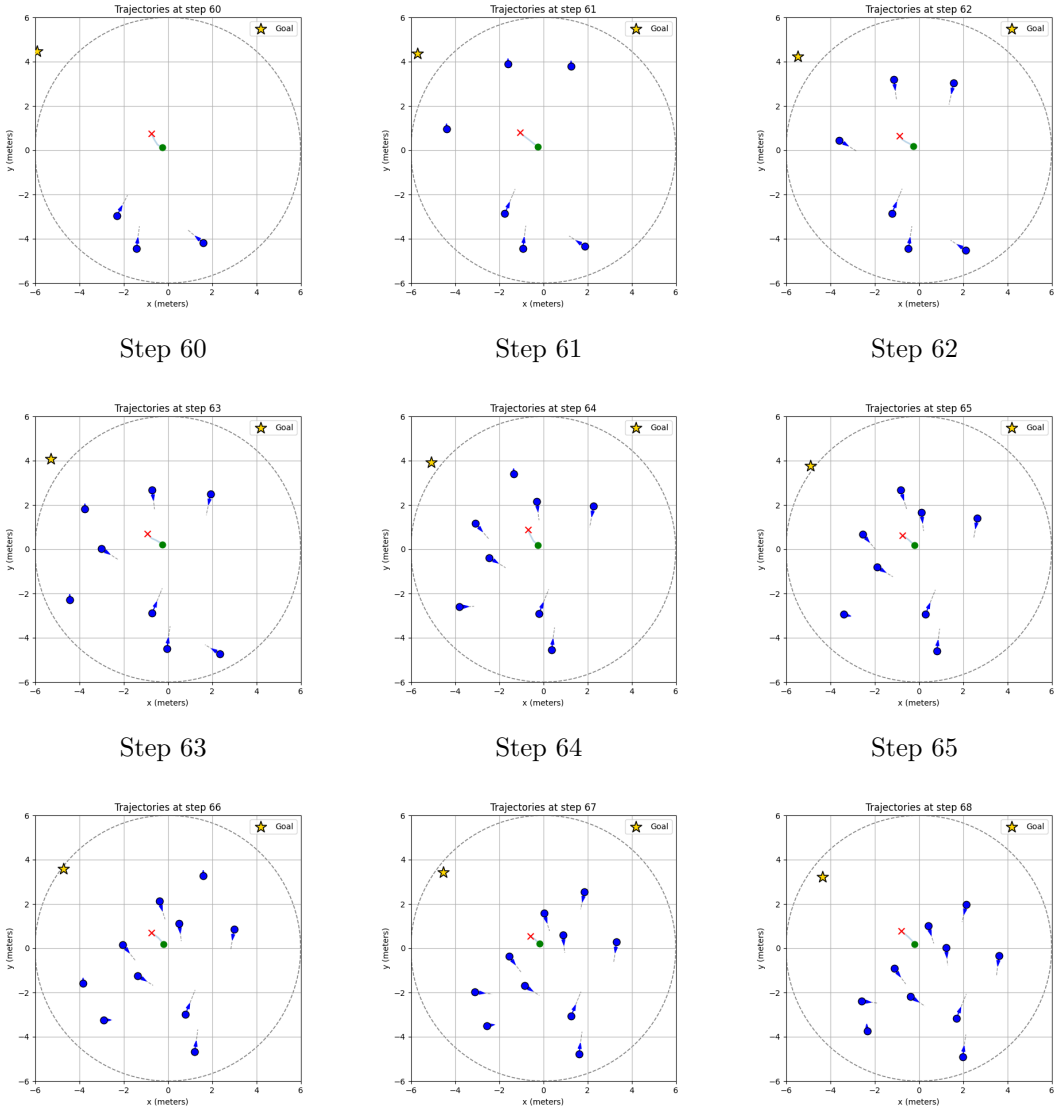
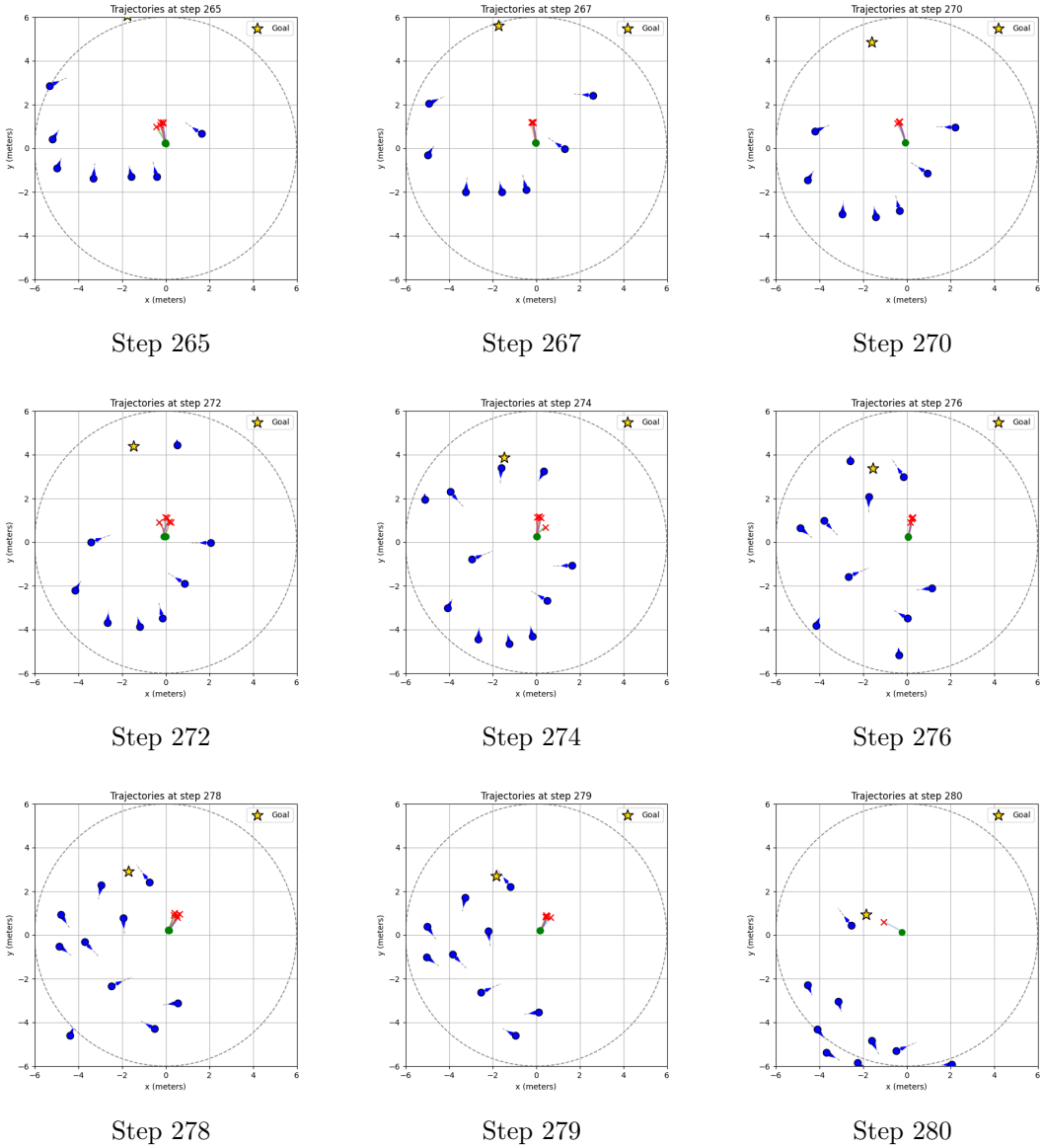


Figure 6.3: Storyboard of a PDPO trajectory from step 60 to step 68. Each snapshot shows the five-step action chunk generated at the current timestep. The sequence illustrates how the policy updates its local plan in response to nearby pedestrians while preserving coherent short-horizon motion.

Figure 6.3 shows a representative episode in which the robot moves toward its goal through a dense crowd. At the beginning of the sequence, the generated chunks point relatively directly toward the target. As nearby humans reduce the available free space, the planned motion shifts toward locally feasible gaps. Several consecutive actions form a turning maneuver whose shape evolves across neighboring replanning steps.

This behavior is particularly visible around steps 62–64. The chunk bends to one side as approaching pedestrians constrain the direct route. Although only the first action is executed, the remaining actions represent the turn as a short sequence. Between steps 65 and 68, the chunks become shorter in the densest interaction region and lengthen again as space opens. The sequence illustrates the combination of planning and feedback: each chunk expresses a local motion intention, while replanning at every timestep adapts that intention to the changing crowd.



Step 265 Step 267 Step 270

Step 272 Step 274 Step 276

Step 278 Step 279 Step 280

Figure 6.4: Multiple action chunks sampled by PDPO under the same local observation. The samples represent different feasible short-horizon motion proposals while remaining compatible with the surrounding crowd geometry.

While Figure 6.3 emphasizes temporal coherence, Figure 6.4 illustrates multimodality. A dense local configuration may permit several reasonable responses, such as moving through different openings, slowing down to yield, or choosing a more conservative heading before resuming progress. The sampled chunks differ in direction and aggressiveness, while each remains a structured short trajectory.

These figures provide qualitative illustrations rather than a quantitative measure of mode coverage. They give a direct view of the object modeled by PDPO: a distribution over local action sequences. Together with the single-action ablation, the examples illustrate how multimodality and temporal coherence are represented jointly in short-horizon trajectory space.

6.6 Scope of the Experimental Evidence

The experiments establish three main empirical findings. Behavioral-cloning initialization improves learning efficiency within the available interaction budget. The five-step action-chunk policy outperforms the corresponding single-action diffusion policy, with the largest difference observed in Bounded CrowdNav. The boundary analysis further shows that workspace handling has a substantial effect on reported navigation success.

The training curves characterize optimization over approximately 12 million environment timesteps. Within this budget, the pretrained policies learn considerably faster than the policy trained from random initialization. The curves do not address the asymptotic performance of direct online training under a much larger interaction budget. The brief decrease at the start of online fine-tuning coincides with the transition from behavioral cloning to policy-induced online data; the present experiments do not further decompose this transient effect.

The action-chunk comparison jointly changes the prediction horizon, the output dimension, and the associated denoising problem. It therefore demonstrates the practical advantage of the five-step formulation as a whole. A comparison over several chunk lengths would provide a finer separation between the effects of temporal coherence, multimodal sequence representation, and optimization.

The qualitative figures illustrate coherent and diverse local motion proposals, but do not measure mode coverage quantitatively. The original and bounded protocols also use different demonstration policies. ORCA generates the demonstrations in the original setting. In Bounded CrowdNav, however, ORCA does not account for the enforced workspace boundary and frequently collides with the wall, which substantially reduces the quality of the resulting demonstrations. We therefore use CrowdNav++ to generate demonstrations that respect the bounded workspace. Comparisons within Bounded CrowdNav use the same environment, evaluation protocol, and test procedure for all learning-based methods.

Chapter 7

Conclusion

This thesis studied diffusion policies for short-horizon planning in robot crowd navigation. The starting point was the observation that dense crowd navigation is not only a problem of selecting an instantaneous collision-free velocity. In many situations, the robot must choose between several qualitatively different local motion patterns, such as passing a pedestrian on either side, slowing down before entering a narrow gap, or making a temporary detour around a group. These alternatives are naturally expressed over several consecutive control steps. This motivates a policy representation that can model distributions over short-horizon action sequences rather than only distributions over single actions.

The thesis first introduced the reinforcement-learning background needed for this formulation. Markov decision processes, value functions, policy-gradient methods, actor–critic estimation, and PPO were reviewed in a way that provides the mathematical basis for the later optimization procedure. The discussion then turned to denoising diffusion models. After deriving the forward noising process, the reverse denoising parameterization, and the noise-prediction objective, the thesis interpreted conditional diffusion models as stochastic policies. This viewpoint makes the generated object an action object: either a single action or an action chunk. The expressiveness discussion showed that a Gaussian policy, even with full covariance over an action chunk, remains a single-mode distribution in the joint action-object space. In contrast, a diffusion policy can represent multimodal conditional distributions over local plans.

Building on this viewpoint, the thesis presented Planning Diffusion Policy Optimization (PDPO) for crowd navigation. PDPO combines offline denoising pretraining with online PPO fine-tuning. In the offline stage, the diffusion policy learns to generate feasible action chunks from demonstration trajectories. In the online stage, the pretrained policy is improved under the environment reward. Since the final action chunk is produced by a multi-step reverse denoising process, the method was formulated through an augmented denoising MDP. This construction separates the outer crowd-navigation MDP from the inner diffusion sampling process and provides a tractable way to apply policy-gradient optimization to the denoising transitions.

The experimental results support the main interpretation of PDPO as a diffusion-based short-horizon planning method. In both the original CrowdNav++ benchmark and the bounded variant, PDPO performs competitively with strong baselines. More importantly, the ablation study shows that the action-chunk formulation is a central part of the improvement: replacing the chunk policy by a single-action diffusion policy substantially weakens performance, especially in the bounded setting. The qualitative visualizations further support this interpretation. The generated chunks evolve coherently across neighboring timesteps and can contain multiple plausible local proposals under the

same observation. Thus, the diffusion policy is not merely an expressive replacement for a Gaussian action distribution; in this setting, it provides a natural representation for multimodal local plans.

The bounded version of the benchmark also highlights the importance of evaluation protocols in robot crowd navigation. If boundary violations are not treated as failures, some trajectories may leave the intended workspace while still being counted as successful. The bounded setting removes this ambiguity and provides a stricter test of whether a policy can navigate through dense local interactions inside the valid domain. This does not make the original benchmark uninformative, since it remains useful for comparison with prior work, but it shows that benchmark details can affect the interpretation of navigation performance.

There are several limitations to the present study. First, all experiments are conducted in simulation. The human agents follow ORCA-based dynamics and do not react to the robot. This enables controlled comparison across methods, but it does not fully capture real human–robot interaction, where pedestrians may adapt their motion in response to the robot. Second, the human trajectory prediction module is deliberately kept simple by using a constant-velocity predictor. This helps isolate the effect of the policy representation, but more realistic navigation systems would require stronger prediction models and a closer integration between prediction and control. Third, diffusion policies require iterative denoising at inference time, which is more expensive than a single forward pass of a standard Gaussian policy. Although the horizon used in this thesis is short, sampling efficiency remains important for real-time robot deployment. Finally, the experiments focus on a specific simulator and a specific crowd-navigation setting. Broader validation across different maps, crowd densities, human behavior models, and physical robots is necessary before drawing strong conclusions about real-world performance.

Several directions follow naturally from this work. One direction is to evaluate diffusion action-chunk policies in environments with more interactive human models, where pedestrians respond to the robot and the robot must reason about mutual adaptation. Another direction is to replace the constant-velocity predictor by a learned trajectory-prediction module, or to train prediction and control jointly. It would also be useful to study more efficient diffusion-policy samplers, such as using fewer denoising steps, distillation, or deterministic sampling schemes, in order to reduce the computational cost of online execution. Finally, the action-chunk horizon itself could be made adaptive: in open regions a longer horizon may encourage efficient progress, while in dense interactions a shorter or more cautious horizon may improve safety and responsiveness.

Overall, the results suggest that diffusion policies are useful for crowd navigation not only because they define expressive stochastic policies, but also because they provide a natural way to model distributions over short-horizon local plans. This perspective connects generative modeling, policy-gradient optimization, and receding-horizon control, and it offers a promising direction for learning navigation policies in dense and multimodal interaction settings.

Appendix A

Implementation Details

This appendix collects implementation details that are useful for reproducing the experiments in Chapter 6. It avoids repeating the main evaluation protocol and focuses on the observation encoder, diffusion-policy network, optimizer settings, model size, checkpoint selection, and training curves.

A.1 Observation Encoder

The observation encoder maps the variable-size crowd observation s_t to a fixed-dimensional latent vector z_t. The input consists of the robot state w_t and a set of human features h_i^t. For each observable human, the feature contains its current position and the predicted positions over the next L timesteps,

$$h_i^t = [u_i^t, \hat{u}_i^{t+1:t+L}], \qquad L = 5. \tag{A.1}$$

The predicted positions are produced by the same constant-velocity predictor used by the learning-based baselines. For batching, the number of humans is padded to a fixed maximum number, and a visibility mask excludes padded or unobservable humans from the attention computation.

Each human feature is first projected by an MLP,

$$e_i^h = \mathrm{MLP}_{\mathrm{human}}(h_i^t). \tag{A.2}$$

Masked multi-head self-attention is then applied over the visible human embeddings to model human–human interactions,

$$(\tilde{e}_1^h, \ldots, \tilde{e}_{n_t}^h) = \mathrm{SelfAttn}(e_1^h, \ldots, e_{n_t}^h). \tag{A.3}$$

The mask ensures that padded entries and unobservable humans do not contribute to the attention weights. Residual connections, layer normalization, and dropout are used after the attention layer.

To obtain robot-conditioned interaction features, the robot state is concatenated with each interaction-aware human embedding and passed through an edge MLP,

$$e_i^{rh} = \mathrm{MLP}_{\mathrm{edge}}([\tilde{e}_i^h, w_t]). \tag{A.4}$$

Layer normalization is applied to the resulting features.

A second attention operation is then applied to the robot-conditioned human features. This allows the representation of each human to be refined using the features of the other humans after the robot state has been incorporated. The query, key, and value features are

$$Q_i = W_Q e_i^{rh}, \qquad K_i = W_K e_i^{rh}, \qquad V_i = W_V e_i^{rh}. \tag{A.5}$$

The corresponding pairwise attention weights are

$$a_{ij} = \frac{\exp\left(Q_i^\top K_j/\sqrt{d_e}\right)}{\sum_{k\in V_t}\exp\left(Q_i^\top K_k/\sqrt{d_e}\right)}, \qquad i,j \in V_t, \tag{A.6}$$

where d_e denotes the edge-feature dimension. The refined robot-conditioned feature of human i is

$$\widetilde{e}_i^{rh} = \sum_{j\in V_t} a_{ij} V_j. \tag{A.7}$$

The resulting human–robot features are aggregated by robot-guided attention pooling. The robot state is first projected into the edge-feature space,

$$e_t^r = W_r w_t. \tag{A.8}$$

For visible humans $i \in V_t$, the attention scores and normalized weights are

$$q_i = (\widetilde{e}_i^{rh})^\top e_t^r, \qquad \alpha_i = \frac{\exp(q_i)}{\sum_{j\in V_t}\exp(q_j)}. \tag{A.9}$$

The pooled crowd representation and final latent vector are

$$e_t^{\text{att}} = \sum_{i\in V_t} \alpha_i \widetilde{e}_i^{rh}, \qquad z_t = \text{MLP}_{\text{latent}}([w_t, e_t^{\text{att}}]). \tag{A.10}$$

Layer normalization is applied before the final latent MLP. The latent vector z_t is used as the conditioning input for the diffusion denoising network and as the input to the value head. For the critic update, z_t is detached so that the value loss does not propagate gradients into the shared observation encoder.

A.2 Diffusion Policy and Value Function

The diffusion actor operates on an action chunk $x_0^t = (x_{0,1}^t, \ldots, x_{0,H}^t) \in \mathbb{R}^{2H}$, where each $x_{0,j}^t \in \mathbb{R}^2$ is a velocity command. In the main PDPO model, $H = 5$, so the policy output dimension is $2H = 10$. For the single-action ablation, the same architecture is used with $H = 1$.

At diffusion step k, the noisy action chunk x_k^t is flattened and concatenated with the observation latent vector z_t and a timestep embedding $\psi(k)$. The timestep embedding is produced by a sinusoidal positional encoding followed by a small MLP. The denoising network predicts the injected Gaussian noise,

$$\widehat{\epsilon} = \epsilon_\theta(x_k^t, z_t, k). \tag{A.11}$$

The output dimension of ϵ_θ matches the flattened action-chunk dimension.

The critic shares the forward representation produced by the observation encoder but does not update the encoder. Writing sg for the stop-gradient operation, the value estimate is

$$V_\varphi(s_t) = \text{MLP}_{\text{value}}\left(\text{sg}(z_t)\right), \qquad z_t = E_\theta(s_t). \tag{A.12}$$

Thus, the actor loss updates the encoder and denoising network through θ, whereas the critic loss updates only the value-head parameters φ. This value estimate is used to compute the environment-level advantage described in Section 5.4.

A.3 Training Hyperparameters

The main training budget and evaluation protocol are described in Section 6.1.4. The tables below list the optimizer, architecture, and diffusion-policy hyperparameters needed for implementation.

Table A.1: Network and diffusion-policy settings.

Hyperparameter	**Value**
Observation dimension	269
Action dimension	2
Action chunk length H	5
Policy output dimension	$2H = 10$
Planning horizon	1.25 s
Denoising network MLP	[512, 256, 256]
Denoising activation	ReLU
Diffusion timestep embedding dimension	16
Value network MLP	[256, 256, 256]
Value activation	Mish
Diffusion objective	noise prediction
Noise schedule	cosine beta schedule
Denoising steps during pretraining	20
Denoising steps during fine-tuning	10

Following DPPO, the implementation uses 20 denoising steps during behavioral cloning and 10 during online fine-tuning to reduce the cost of rollout collection.

Table A.2: Behavioral-cloning pretraining hyperparameters.

Hyperparameter	**Value**
Optimizer	AdamW
Batch size	128
Learning rate	1×10^{-3}
Weight decay	1×10^{-6}

Table A.3: Online PPO fine-tuning hyperparameters.

Hyperparameter	**Value**
Optimizer	AdamW
Actor learning rate	1×10^{-4}
Critic learning rate	1×10^{-3}
Discount factor γ	0.99
GAE parameter λ	0.95
Denoising discount factor γ_{denoise}	0.99
PPO batch size	500
PPO update epochs	5
Value loss coefficient	0.5
Denoising policy loss clipping coefficient	0.01
Base denoising policy loss clipping coefficient	0.001
Denoising policy loss clipping growth rate	3
Target KL	1.0

The denoising policy loss clipping coefficient is used in the clipped diffusion policy loss during online fine-tuning. It is distinct from the PPO probability ratio clipping parameter introduced in Definition 2.34.

A.4 Model Size and Checkpoint Selection

During online fine-tuning, PDPO updates 1.049M trainable parameters, consisting of a 0.848M-parameter diffusion actor, including the shared observation encoder, and a 0.201M-parameter critic head. Under our implementation, CrowdNav++ contains 2.503M trainable parameters. Thus, the improvement of PDPO in Chapter 6 is not due to using a larger neural network.

Table A.4: Trainable parameter count of learning-based methods.

Method	**Trainable parameters**
CrowdNav++ (Const. Vel.)	2.503M
PDPO (Ours)	1.049M

For final test evaluation, we select the checkpoint with the highest success rate on validation episodes during online fine-tuning. The selected checkpoint is then evaluated on the held-out test seeds used for the results in Chapter 6.

A.5 Additional Training Curves

Figure A.1 shows the online fine-tuning curves for PDPO, the training from scratch ablation and the single-action diffusion-policy ablation on Bounded CrowdNav. These curves are used as auxiliary diagnostics; the main performance comparison is based on the final evaluation reported in Sections 6.3 and 6.4.

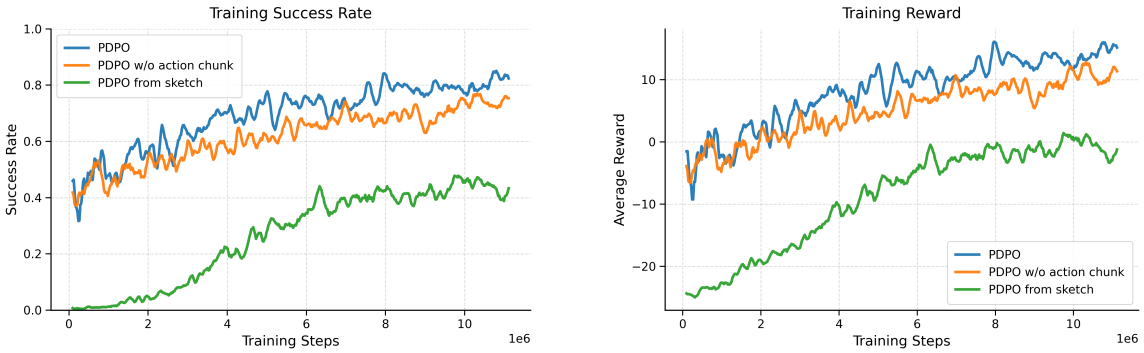


Figure A.1: Online fine-tuning curves of PDPO, the training from scratch ablation and the single-action ablation on Bounded CrowdNav. Left: training success rate. Right: average training reward. The curves provide an auxiliary view of the optimization process and are not used as a substitute for final test evaluation.

Bibliography

[Aja+23] A. Ajay, Y. Du, A. Gupta, J. B. Tenenbaum, T. S. Jaakkola and P. Agrawal. 'Is Conditional Generative Modeling All You Need for Decision-Making?' In: *International Conference on Learning Representations (ICLR)*. 2023.

[Bel57] R. Bellman. *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[Ber+11] J. van den Berg, S. J. Guy, M. Lin and D. Manocha. 'Reciprocal n-Body Collision Avoidance'. In: *Robotics Research*. Vol. 70. Springer Tracts in Advanced Robotics. Springer, 2011, pp. 3–19.

[Bis94] C. M. Bishop. *Mixture Density Networks*. Tech. rep. NCRG/94/004. Birmingham, UK: Aston University, 1994.

[Che+19] C. Chen, Y. Liu, S. Kreiss and A. Alahi. 'Crowd-Robot Interaction: Crowd-Aware Robot Navigation with Attention-Based Deep Reinforcement Learning'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 6015–6022.

[Che+20] C. Chen, S. Hu, P. Nikdel, G. Mori and M. Savva. 'Relational Graph Learning for Crowd Navigation'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 10007–10013.

[Chi+25] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake and S. Song. 'Diffusion Policy: Visuomotor Policy Learning via Action Diffusion'. In: *The International Journal of Robotics Research* 44.10–11 (2025), pp. 1684–1704.

[Eve+18] M. Everett, Y. F. Chen and J. P. How. 'Motion Planning among Dynamic, Decision-Making Agents with Deep Reinforcement Learning'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 3052–3059.

[Gre+04] E. Greensmith, P. L. Bartlett and J. Baxter. 'Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning'. In: *Journal of Machine Learning Research* 5 (2004), pp. 1471–1530.

[HM95] D. Helbing and P. Molnár. 'Social Force Model for Pedestrian Dynamics'. In: *Physical Review E* 51.5 (1995), pp. 4282–4286.

[Ho+20] J. Ho, A. N. Jain and P. Abbeel. 'Denoising Diffusion Probabilistic Models'. In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 6840–6851.

[Jan+22] M. Janner, Y. Du, J. B. Tenenbaum and S. Levine. 'Planning with Diffusion for Flexible Behavior Synthesis'. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 9902–9915.

[KT00] V. R. Konda and J. N. Tsitsiklis. 'Actor-Critic Algorithms'. In: *Advances in Neural Information Processing Systems*. Vol. 12. 2000, pp. 1008–1014.

[Liu+21] S. Liu, P. Chang, W. Liang, N. Chakraborty and K. Driggs-Campbell. 'Decentralized Structural-RNN for Robot Crowd Navigation with Deep Reinforcement Learning'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 3517–3524.

[Liu+23] S. Liu, P. Chang, Z. Huang, N. Chakraborty, K. Hong, W. Liang, D. L. McPherson, J. Geng and K. Driggs-Campbell. 'Intention Aware Robot Crowd Navigation with Attention-Based Interaction Graph'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 12015–12021.

[Put94] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[Ren+25] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai and M. Simchowitz. 'Diffusion Policy Policy Optimization'. In: *International Conference on Learning Representations (ICLR)*. 2025.

[SB18] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2nd. MIT Press, 2018.

[Sch+16] J. Schulman, P. Moritz, S. Levine, M. I. Jordan and P. Abbeel. 'High-Dimensional Continuous Control Using Generalized Advantage Estimation'. In: *International Conference on Learning Representations (ICLR)*. 2016.

[Sch+17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov. 'Proximal Policy Optimization Algorithms'. In: *arXiv preprint arXiv:1707.06347* (2017). arXiv: 1707.06347 [cs.LG].

[Shi+22] W. Shi, Y. Zhou, X. Zeng, S. Li and M. Bennewitz. 'Enhanced Spatial Attention Graph for Motion Planning in Crowded, Partially Observable Environments'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2022, pp. 4750–4756.

[Soh+15] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan and S. Ganguli. 'Deep Unsupervised Learning Using Nonequilibrium Thermodynamics'. In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. PMLR, 2015, pp. 2256–2265.

[Sut+99] R. S. Sutton, D. A. McAllester, S. P. Singh and Y. Mansour. 'Policy Gradient Methods for Reinforcement Learning with Function Approximation'. In: *Advances in Neural Information Processing Systems*. Vol. 12. 1999, pp. 1057–1063.

[Sut88] R. S. Sutton. 'Learning to Predict by the Methods of Temporal Differences'. In: *Machine Learning* 3.1 (1988), pp. 9–44.

[TK10] P. Trautman and A. Krause. 'Unfreezing the Robot: Navigation in Dense, Interacting Crowds'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010, pp. 797–803.

[WD92] C. J. C. H. Watkins and P. Dayan. 'Q-Learning'. In: *Machine Learning* 8.3–4 (1992), pp. 279–292.

[Wil92] R. J. Williams. 'Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning'. In: *Machine Learning* 8.3–4 (1992), pp. 229–256.

[Yao+25] J. Yao, X. Zhang, Y. Xia, Z. Wang, A. Roy-Chowdhury and J. Li. 'Towards Generalizable Safety in Crowd Navigation via Conformal Uncertainty Handling'. In: *Proceedings of the 9th Conference on Robot Learning*. Vol. 305. Proceedings of Machine Learning Research. PMLR, 2025, pp. 4206–4225.

[ZG24] Y. Zhou and J. Garcke. 'Learning Crowd Behaviors in Navigation with Attention-Based Spatial-Temporal Graphs'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 5485–5491.