Reinforcement Learning-based Local Motion Planning for Mobile Robots

Ng Wen Zheng Terence

College of Computing and Data Science

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy (Ph.D)

2025

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

22/07/2024
. . . . . .
Date

. . . . . .
Ng Wen Zheng Terence

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

22/07/2024

. . . . . .
Date

. . . . . .
Prof. Zhang Tianwei

Authorship Attribution Statement

Please select one of the following; *delete as appropriate:

(B) This thesis contains material from 4 paper(s) published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Please amend the typical statements below to suit your circumstances if (B) is selected.

Chapter 3 is published as **Wen Zheng Terence Ng, Jianda Chen, 'Dual Action Policy for Robust Sim-to-Real Reinforcement Learning'. In '33rd International Conference on Artificial Neural Networks (ICANN)', 2024**.

The contributions of the co-authors are as follows:

- Chen Jianda was involved in discussions about the methodology and experiments.

Chapter 4 is published as **Wen Zheng Terence Ng, Jianda Chen, Tianwei Zhang, 'Off-dynamics Conditional Diffusion Planners'. In '2024 IEEE International Conference on Intelligent Robots and Systems (IROS)'**.

The contributions of the co-authors are as follows:

- Prof Zhang Tianwei guided the initial research direction and revised the manuscript drafts.
- Chen Jianda was involved in discussions about the methodology and experiments.

Chapter 5 is published as **Wen Zheng Terence Ng, Jianda Chen, Tianwei Zhang, 'Latent Embedding Adaptation for Human Preference Alignment in Diffusion Planners'. In '2025 IEEE International Conference on Robotics and Automation (ICRA)'**.

The contributions of the co-authors are as follows:

- Prof Zhang Tianwei guided the initial research direction and revised the manuscript drafts.
- Chen Jianda was involved in discussions about the methodology and experiments.

Chapter 6 is published as **Wen Zheng Terence Ng, Jianda Chen, Sinno Jialin Pan, Tianwei Zhang, 'Improving the Generalization of Unseen Crowd Behaviors for Reinforcement Learning based Local Motion Planners'. In '2024 IEEE International Conference on Robotics and Automation (ICRA)'**.

The contributions of the co-authors are as follows:

- Prof Zhang Tianwei guided the initial research direction and revised the manuscript drafts.
- Prof Sinno Jialin Pan and Chen Jianda was involved in discussions about the methodology and experiments.

22/07/2024
. . . . . .
Date

. . . . . .
[Input Name Here & Sign Above]

Abstract

This thesis explores data-driven approaches, particularly reinforcement learning (RL), for enhancing the autonomous navigation capabilities of mobile robots in complex and dynamic environments. While RL-based motion planners have shown promise in various scenarios, they encounter two major challenges: the reality gap and human-centered design. The reality gap, a discrepancy between data collected from offline sources or simulations and the real-world environment, poses a major challenge for transferring learned behaviors to real-world deployment. Additionally, ensuring human-centered design is crucial for tailoring robot behavior to individual preferences and adhering to social norms, which are essential for successful integration into human environments. This thesis focuses on addressing both challenges to improve the practicality and broader adoption of RL-based local motion planners.

In the first part of the thesis, we address the dynamics mismatch inherent in both online and offline RL. In the online RL setting, dynamics mismatch arises when the agent learns through interaction with a simulated environment that does not fully represent real-world dynamics. To address this, we introduce the Dual Action Policy (DAP), a novel method that uses a single policy to predict two sets of actions: one for maximizing task rewards in simulation and another for adapting to real-world dynamics through reward adjustments. This decoupling approach simplifies the optimization landscape, leading to more effective and robust policy convergence. Our experiments demonstrate that DAP significantly improves baseline performance using only a small amount of data from the target environment. In the offline RL setting, we tackle the dynamics mismatch that occurs when learning from large-scale datasets that do not perfectly match the target environment's dynamics. Our approach utilizes conditional diffusion models to learn a joint distribution between a readily available off-dynamics source dataset and a limited target dataset. We also incorporate a continuous dynamics score and an inverse-dynamics context to better capture the underlying dynamics structure and generate trajectories that align

with the target environment's dynamics. This approach enhances adaptation and robustness to subtle dynamic shifts in the target domain.

The second part of this work focuses on human-centered solutions for RL-based planners. First, to tackle the challenge of personalizing robot behaviors, we introduce a resource-efficient approach that enables rapid adaptation to individual user preferences. Our method leverages a pretrained conditional diffusion model with preference latent embeddings (PLE), trained on a large, reward-free offline dataset. The PLE serves as a compact representation for capturing specific user preferences. By adapting the pretrained model using our proposed preference inversion method, which directly optimizes the learnable PLE, we achieve superior alignment with human preferences compared to existing solutions. Finally, we address the challenge of navigating crowded environments with unpredictable pedestrians. To tackle this problem, we introduce an efficient method that enhances agent diversity within a single policy by maximizing an information-theoretic objective. This diversity enriches each agent's experiences, leading to improved adaptability to unseen crowd behaviors. Our behavior-conditioned policies outperform existing methods in various scenarios inspired by pedestrian crowd behaviors, effectively reducing potential collisions without increasing travel time or distance..

In summary, this thesis presents novel solutions to tackle the reality gap and incorporates human-centered design principles, advancing the applicability and acceptance of RL-based local motion planners. The research presented here marks a substantial step forward in bridging the gap between theoretical RL models and their real-world applications, extending beyond motion planning and robotics to impact a broader range of intelligent systems and autonomous technologies.

Acknowledgments

This PhD journey would not have been possible without the support, guidance, and encouragement of many individuals who have been integral in shaping both my research and my personal growth. I would like to take this opportunity to acknowledge their contributions and express my heartfelt appreciation.

I would like to express my deepest gratitude to my supervisor, **Prof. Zhang Tianwei**, for his unwavering guidance and support throughout my PhD journey. His profound expertise, insightful advice, and constant encouragement have been instrumental in shaping my research. I am grateful for his patience, trust, and motivation, which pushed me to reach my full potential. His role as both a mentor and a role model has inspired me in many aspects of life, and I am truly grateful for the opportunity to learn from him.

I would like to express my sincere gratitude to **Prof. Sinno Pan** for his invaluable guidance during my initial two years of PhD studies. His extensive knowledge and mentorship allowed me to develop a broad technical foundation and fostered my growth as an independent researcher. I am deeply grateful to my main collaborator and senior, **Dr. Chen Jianda**, for the countless hours of insightful discussions on various aspects of reinforcement learning. His willingness to share his experiences and knowledge helped me navigate through challenges and setbacks. I extend my thanks to the following individuals who contributed to my research through engaging technical discussions: **Shi Haosen, Chen Guan Lin, and Dr. Michael Hoy**. I would thank the **Economic Development Board of Singapore** for their generous scholarship and funding, which made this research possible.

I am grateful to my colleagues at Continental Automotive: **Vinay Adiga** and **Dr. David Woon**, who encouraged me to embark on this PhD journey; **Eric Juliani**, for his continuous support during the challenging years; **Andreas Hartmannsgruber** and **Vincent Wong**, for providing me with the space and support to focus on my research; **Dr. Shen Ren** for being

an awesome teammate and always having my back; my **entire AI team** for their encouragements throughout these years; the support staff from the Conti-NTU Corp Lab: **Dr. Lee Meng Yeong** and **Ethel Sng** with legal matters. I would like to thank my friends who supported and encouraged me along the way: **Dr. Ma Yu Kun, Dimitris Geromichalos, Matteo Pelati, Erin Chang, Yeoh Oon Jie, and Kim Seol Hee**. I am grateful to my former colleagues, **Dr. Tran Huy Dat** and **Dr. Jonathan Dennis**, who ignited my passion for machine learning. I extend my heartfelt gratitude to my family members for their constant support: my **parents** and **parents-in-law**, my **siblings** and my uncle **Raymond Ang**.

To the most important people in my life: my wife, **Chen Yi Jie**, thank you for your immense sacrifices, understanding, encouragement, and trust. I could not have achieved this without you. Thank you for taking this path and walking this journey with me. *I love you*. To my children, **Charlotte Ng** and **Owen Ng**, thank you for teaching me to appreciate the simplest things in life. You are growing up in the most exciting times of AI and Robotics; and will witness incredible advancements. I cannot wait to see how these technologies shape your world. I promise to make up for all the lost time and share in your journey of discovery. Finally, to **myself**: This has been by far the toughest challenge, juggling work, PhD, and raising two young children. Although difficult, this journey has been incredibly fruitful and satisfying. As this chapter of my academic journey comes to a close, I am reminded that the pursuit of knowledge is endless, beautifully captured in Isaac Newton's words:

> "I do not know what I may appear to the world, but to myself, I seem to have been only like a boy playing on the seashore, diverting myself now and then in finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me."

To my dear family.

Contents

List of Figures

List of Tables

List of Abbreviations

A2C	Advantage Actor-Critic
CQL	Conservative Q-Learning
CTDE	Centralized Training with Decentralized Execution
DAP	Dual Action Policy (Proposed)
DARC	Domain Adaptation with Rewards from Classifier
DARA	Dynamics-Aware Reward Augmentation
DDGP	Deep Deterministic Policy Gradient
DIAYN	Diversity is All You Need
DNN	Deep Neural Networks
DPO	Direct Policy Optimization
DPM	Diffusion Probabilistic Models
DQN	Deep Q-Networks
DR	Domain Randomization
DT	Decision Transformer
DWA	Dynamic Window Approach
FID	Frechet Inception Distance
GAIL	Generative Adversarial Imitation Learning
GAN	Generative Adversarial Network
GARAT	Generative Adversarial Reinforced Action Transformation
GAT	Grounded Action Transforms
IN	Invisible (Proposed)
IPO	Identity Preference Optimization
KL	Kullback-Leibler
LiDAR	Light, Detection and Ranging
LSTM	Long Short-Term Memory
MC	Monte-Carlo
MDP	Markov Decision Process
MSE	Mean Squared Error
NH	Non-Homogeneous (Proposed)
NLP	Natural Language Processing
OOD	Out-of-distribution
ORCA	Optimal Reciprocal Collision Avoidance

PPO	Proximal Policy Optimization
PLE	Preference Latent Embedding (Proposed)
PT	Preference Transformers
QD	Quality Diversity
RGB	Red, Green, Blue
RGAT	Reinforced Grounded Action Transforms
RL	Reinforcement Learning
RLHF	Reinforcement Learning from Human Feedback
SAC	Soft Actor-Critic
SF	Social Force (Proposed)
SO	Sub-optimal (Proposed)
TD	Temporal Difference
TEB	Timed Elastic Band
TT	Trajectory Transformer
VA	Variability (Proposed)
VO	Velocity Obstacle (Proposed)

List of Notations

$\mathcal{M}$	Markov Decision Process
$\mathcal{S}$	State space
$\mathcal{A}$	Action space
$\mathcal{R}$	Reward function
P	Transition distribution
γ	Discount factor
π	Policy
s	State
a	Action
r	Reward signal
V	State value function
Q	State-action value function
$\mathcal{D}$	Dataset (General)
$\mathcal{H}$	Entropy
τ	Trajectory
H	Horizon length
$\mathbb{D}_{\text{KL}}$	Kullback–Leibler divergence
$q(x)$	Forward diffusion process
$p_\theta(x)$	Reverse diffusion process
K	Number of diffusion steps
ϵ	Diffusion noise
$\alpha_{0:K}$	Diffusion noise schedule
$\mathcal{D}_{\text{source}}$	Dataset from source MDP
$\mathcal{D}_{\text{target}}$	Dataset from target MDP
κ	Scaling coefficient for Dynamics score context
x^w	Winning sample from preference pair
x^l	Losing sample from preference pair
β	KL penalty coefficient
z	Preference latent embedding
z^w	Winner's Preference latent embedding
z^l	Loser's Preference latent embedding
h	Behavior tokens

M	Number of behavior tokens
ζ	Mean KL div between pairwise action distribution

List of Publications

International Conferences

1. Wen Zheng Terence Ng, Jianda Chen, Sinno Jialin Pan, Tianwei Zhang, "Improving the Generalization of Unseen Crowd Behaviors for Reinforcement Learning based Local Motion Planners". In "IEEE International Conference on Robotics and Automation (ICRA)", Yokohama, Japan, 2024.

2. Wen Zheng Terence Ng, Jianda Chen, "Dual Action Policy for Robust Sim-to-Real Reinforcement Learning". In "33rd International Conference on Artificial Neural Networks (ICANN)", Lugano, Switzerland, 2024.

3. Wen Zheng Terence Ng, Jianda Chen, Tianwei Zhang, "Off-dynamics Conditional Diffusion Planners". In "IEEE International Conference on Intelligent Robots and Systems (IROS)", Abu Dhabi, UAE, 2024.

4. Wen Zheng Terence Ng, Jianda Chen, Tianwei Zhang, "Latent Embedding Adaptation for Human Preference Alignment in Diffusion Planners". In "IEEE International Conference on Robotics and Automation (ICRA)", Atlanta, USA 2025.

International Workshops

1. Wen Zheng Terence Ng, Jianda Chen, Kangjie Chen, Zichen Chen, Tianwei Zhang, "Personalizing Diffusion Planners with Efficient Preference Optimization". In "International Conference on Learning Representations Workshop (ICLR Workshop)", Vienna, Austria, 2024.

Chapter 1

Introduction

1.1 Background

Motion planning, the process of determining a sequence of valid geometric configurations that guide an object from its initial state to a desired goal state, is a fundamental problem in robotics [2]. This field has attracted significant research interest over the years, leading to the development of numerous approaches for solving this complex problem. In the specific context of wheeled mobile robots, motion planning translates to finding a collision-free path that allows the robot to move from its starting position to a designated goal position. While humans effortlessly navigate through their surroundings, achieving robust and reliable motion planning for autonomous mobile robots remains a considerable challenge.

Motion planning algorithms can be broadly classified into two categories: global and local motion planning [3, 4]. Both types of planning share the same objective of finding a path from one point to another, but their approaches differ significantly. Global motion planning operates under the assumption of a known and static environment, often relying on a map or a priori information to generate a complete path from start to finish. Local motion planning, on the other hand, does not assume any prior knowledge of the environment. Instead, it relies on real-time sensor data, such as LiDAR or RGB camera input, to perceive the immediate surroundings and dynamically adjust the robot's path to avoid obstacles and respond to changes in the environment. Our research primarily focuses on this local motion planning aspect.

The ability of mobile robots to maneuver and navigate in complex and dynamic environments, particularly through effective local motion planning, is essential for their successful integration into various real-world applications [5]. In industrial automation, robots can work

alongside humans in factories and warehouses, optimizing workflows and increasing productivity. In service delivery, these robots can autonomously navigate through crowded urban spaces to deliver packages, groceries, or food, improving convenience and efficiency. In agriculture, robots can traverse fields, orchards, and vineyards to perform tasks such as harvesting, spraying, and monitoring crops, reducing manual labor and increasing yields. Additionally, local motion planning is crucial for robots operating in public spaces, such as shopping malls, airports, or hospitals, where they need to interact with people safely and efficiently. By continuously sensing and adapting to their surroundings, local motion planners ensure that robots can avoid collisions, navigate around obstacles, and safely reach their destinations, even in the face of unexpected events or changes in the environment.

Classical local motion planning algorithms offer a foundation for addressing the challenges of navigating complex and dynamic environments. These algorithms, often based on geometric principles and heuristics, provide a structured way to compute collision-free trajectories for mobile robots. Common examples include the Dynamic Window Approach (DWA) [6], which optimizes the robot's velocity profile to avoid obstacles while maximizing progress towards the goal, and the Timed Elastic Band (TEB) algorithm [7], which deforms an initial trajectory to accommodate dynamic obstacles while maintaining smoothness. However, classical methods have several limitations [8, 9]. They can become computationally expensive and less efficient when higher accuracy is required through increased complexity. They may also struggle in noisy environments due to sensor inaccuracies, as they heavily rely on the perception module. Compounded errors in the perception module can negatively impact the performance of classical methods compared to end-to-end systems, which directly utilize raw sensor values for decision-making. Additionally, their reliance on pre-defined rules and heuristics makes them less adaptable to unexpected situations and novel obstacles. It is particularly difficult to incorporate social aspects of navigation, such as understanding and predicting pedestrian behavior, into these methods compared to data-driven approaches that can learn these nuances from real-world data.

Model Predictive Control (MPC) offers a sophisticated approach by optimizing control inputs through finite-horizon optimization at each timestep, considering system dynamics, constraints, and future predictions [10]. While MPC effectively models robot dynamics and handles various constraints, it faces significant challenges in dynamic environments with human

pedestrians [11–13]. Modeling pedestrian behavior is particularly difficult due to its unpredictability, variability, and complex social interactions. Furthermore, MPC's performance is highly sensitive to the planning horizon length, and achieving optimal performance often requires substantial computational resources at runtime. This computational burden increases with environmental complexity, such as in crowded spaces with multiple pedestrians. Additionally, the effectiveness of MPC depends heavily on the accuracy of its predictive models, which becomes challenging when accounting for the inherent uncertainty and variability of human behavior. In crowded environments, these modeling challenges, combined with computational demands, can lead to reduced performance or delayed responses.

In recent years, data-driven approaches, particularly reinforcement learning (RL) [14], have gained traction in addressing the autonomous navigation problem for mobile robots [15–26]. Unlike classical methods that rely on heuristics, RL enables robots to learn navigation policies through experience, making them more adaptable to complex and dynamic environments. The RL framework treats the robot as an agent that interacts with its environment. By receiving sensor data as input, the agent makes decisions about actions like steering and speed, and then receives feedback in the form of rewards or penalties based on the outcome of those actions. Through repeated trial and error, the robot learns to optimize its behavior to maximize long-term rewards, resulting in efficient and safe navigation. Several key advantages make RL particularly well-suited for local motion planning. Firstly, by eliminating the need for manually designed rules, RL-based approaches are inherently more robust and adaptable to unforeseen situations and obstacles. Secondly, RL can learn directly from raw sensor data, reducing the dependency on the performance of the perception modules. Thirdly, RL-based methods can continuously learn and improve with experience, adapting to changes in the environment or the robot's own capabilities. This adaptability is crucial for real-world deployment, where conditions are rarely static. Lastly, RL offers a framework for incorporating social aspects of navigation, such as understanding and predicting pedestrian behavior, by learning from demonstrations or interactions with humans.

RL-based planners have already demonstrated their effectiveness in various challenging scenarios, outperforming classical methods in cluttered and dynamic environments. The flexibility of RL allows for the incorporation of diverse objectives into the reward function, including social norms, smoother motion, and safety constraints. Furthermore, RL has the potential

to effectively handle high-dimensional state spaces and complex sensor inputs, which can be challenging for classical planning methods. Despite their successes, RL-based planners still face significant challenges in real-world applications. The literature review highlights two key areas that require attention: the reality gap and human-centered design.

1. The reality gap in RL, often referred to as the *sim-to-real* gap in the field of robotics, is the discrepancy between the performance of models trained in simulation and their performance in real-world environments [27–29]. In the RL-based planner literature, most research trains and tests these planners in simulated environments, leaving their real-world effectiveness uncertain due to this gap. While some researchers attempt to bridge this gap by incorporating safety constraints into reward shaping, these approaches often lack generalizability to diverse real-world situations. Addressing the reality gap is essential for ensuring the safe and effective operation of mobile robots in real-world applications. This gap also extends beyond training in simulation, even when offline data collected from other sources is being used for training.

2. Human-centered design in RL-based planners is another key challenge, with two main areas encompassing the handling of human pedestrian behavior and the personalization of robot behavior. First, understanding and predicting pedestrian behavior is crucial for mobile robots to navigate in human populated environments [19]. This allows robots to anticipate movements, avoid collisions, and plan efficient paths, ensuring safety and smooth operation. Although some work has considered social aspects and human-robot interaction in reward design, these socially-aware planners may fail when faced with the unpredictable movements of real-world crowds. Second, personalization of mobile robots is essential for enhancing user experience, improving safety, and gaining trust and acceptance [30, 31]. By tailoring a robot's navigation style to individual preferences, it can navigate more efficiently, safely and predictably, thereby increasing its overall usability and impact. This adaptability can also lead to greater user satisfaction and promote wider adoption of robotic technology in various domains. While personalization has been extensively studied in other fields like Natural Language Processing (NLP) [32], it remains less explored in sequential decision-making models.

Closing the reality gap and implementing human-centered approaches are closely interconnected in advancing RL-based planners. Addressing the reality gap, by improving sim-to-real transfer and ensuring robustness to discrepancies in offline data, ensures that planners operate reliably in environments that closely approximate real-world dynamics. This reliability is essential for enabling human-centered features such as socially compliant navigation and personalized behaviors, which require consistent performance in dynamic, human-populated settings. For instance, planners that overcome sim-to-real gaps can more effectively anticipate pedestrian movements, while personalization frameworks leverage this reliability to tailor navigation styles to individual user preferences. Together, these efforts form a cohesive foundation: closing the reality gap ensures functional robustness, while human-centered design provides adaptability to human needs. By unifying these efforts, this work underscores how RL-based planning must balance technical robustness with human adaptability, paving the way for systems that are both reliable and socially intelligent, and ultimately enabling their practical applicability.

1.2 Motivation

1.2.1 Bridging the Reality Gap for RL-based Planners

In RL-based motion planners, to enable the agent to interact and learn within an environment, most research has primarily relied on simulations to validate and verify proposed ideas [33–36]. Training the RL agent in a simulator offers a safe environment without the risk of damaging physical robots and obstacles. It is also cost-effective since no physical hardware needs to be purchased or designed. Most importantly, simulators allow data collection much faster than real-time. However, deploying agents trained in a simulator directly to the real world often suffers from the sim-to-real gap. This gap is a long-standing problem in the field of robotics and has been the subject of extensive research.

This gap mainly derives from two primary sources: sensor mismatch and dynamics mismatch. (1) Sensor mismatch arises when simulators fail to accurately reproduce textures, shapes, and lighting conditions found in the real world [29]. This discrepancy causes data-driven models, which are often sensitive to such distribution shifts, to fail. For RL-based planners, most work relies on 2D laser scanners as the main sensor to detect obstacles in their

surroundings. The 2D laser scanner returns a one-dimensional array of distance measurements. Its main advantage lies in the relatively low distribution shift when transitioning from simulation to real-world deployment, making it less susceptible to the sim-to-real gap in our scenario. (2) Dynamics mismatch often arises from the inability to perfectly replicate real-world physics [27]. Discrepancies in, for example, lag time, inertia, and friction contribute to this issue. Simulated robots might respond instantaneously, while real robots experience delays. Inertia effects, often simplified in simulations, can lead to unrealistic acceleration and deceleration profiles. Additionally, neglecting joint and gear frictions, which significantly impact real-world robot movement, further exacerbates the mismatch. These discrepancies collectively affect the robot's performance, which can become problematic if not properly addressed. In the context of RL-based planners, some proposed methods attempt to circumvent the problem without directly addressing the mismatch. For example, to mitigate the effects of dynamics mismatch, several approaches propose adding safety constraints as a conservative strategy or employing hybrid control strategies. Also, most work primarily evaluated their methods in simulation without considering the mismatch, and this lack of proven reliability in deploying these robots in the real world has been highlighted in several review papers.

An alternative to simulation-based training is *offline RL*, which uses data collected from real robots [37]. This method is safe and cost-effective, potentially avoiding some sim-to-real issues like dynamics mismatch. However, even with data from matching robots, slight differences in dynamics can exist between individual units due to manufacturing variations, wear and tear, or environmental factors. This can lead to some dynamics mismatch, affecting the performance of RL agents when transferred to new robots. Interestingly, allowing the use of robots with slightly different dynamics for offline data collection can be more cost-effective. This approach reduces the need to maintain high precision similarity among robots and increases data availability. By addressing dynamics mismatch in offline RL, it opens up the possibility of using robots with slightly different dynamics for offline data collection. It reduces the need to maintain high precision similarity among robots and increases data availability, potentially offering more realistic and versatile training for robotic systems. The dynamics mismatch remains a significant challenge in both online (simulation-based) and offline RL approaches. Addressing this issue is crucial for improving the reliability and performance of RL agents when deployed in real-world environments, regardless of the training method used.

1.2.2 Handling Human Pedestrians Behavior

RL-based motion planning relies heavily on the training environment to develop effective decision-making capabilities. A critical aspect is accurately representing diverse and unpredictable human pedestrian movements, influenced by personal preferences, social norms, and environmental factors. To learn a robust policy, the RL agent's environment must capture a wide range of pedestrian behaviors. Two main approaches exist for generating these movements: single-agent methods and decentralized multi-agent collision avoidance. Single-agent approaches include assigning waypoints from datasets, manually designing behaviors, or using fixed algorithms, but may lack diversity and lead to overfitting [18, 21]. In contrast, decentralized multi-agent approaches, where agents learn to navigate and avoid collisions independently, avoid potential design biases [15, 17, 19]. They also have greater sample efficiency as all agents can contribute to the learning process. However, one drawback is that under this multi-agent framework, agents tend to behave homogeneously, which may struggle to generalize to diverse and unseen crowd behaviors. Most studies evaluate their proposed RL-based planners using the same pedestrian scenarios as in training. This approach lacks robust testing in diverse, realistic conditions. As a result, there is insufficient evidence of these robots' reliability when deployed in complex environments with varied pedestrian behaviors. This remains a significant challenge in the field of RL-based motion planning for robots interacting with humans.

1.2.3 Personalization for RL-based Planners

Numerous studies have explored reward shaping techniques to influence how RL-based agents navigate obstacles and reach goals. These reward structures primarily aim to enhance human-robot interaction and improve safety in shared environments. One of the earliest and most intuitive approaches to improve human-robot interaction involves incorporating social norms into the reward function. For example, researchers have introduced rewards for behaviors such as not overtaking or cutting off others, giving way to pedestrians, and maintaining appropriate social distances when passing [19]. These socially-aware reward structures encourage robots to navigate in a manner more aligned with human expectations and comfort levels. Another common aspect of reward shaping focuses on adding safety guidance to the training process [23]. This approach encourages more conservative robot behavior, preparing for worst-case scenarios to prevent collisions. Safety-oriented rewards might penalize close proximity to obstacles

or reward maintaining a safe distance from pedestrians. Researchers have also proposed methods to improve the smoothness of robot motion through reward shaping [17]. Rewards for smooth motion might penalize sudden accelerations or changes in direction. Smoother trajectories can reduce mechanical wear and tear, improve energy efficiency, and create more predictable movements. This predictability is crucial for enhancing safety for nearby humans, as it allows them to anticipate the robot's actions more easily.

However, a major challenge with personalizing through reward shaping is the risk for human bias and limited generalization. For instance, a robot trained to prioritize maintaining large personal space might inconvenience people in crowded areas who are used to closer proximity. Social norms can vary greatly between individuals and cultures, making it difficult to design a reward function that suits everyone. Moreover, over-prioritizing smoothness in robot motion can lead to decreased efficiency, as the robot may sacrifice speed or take longer paths to avoid abrupt movements. Instead of manually designing these rewards, a more effective approach might involve rapid and accurate customization of the robot's behavior based on individual preferences.

1.3 Thesis Objective

Our main objective is to enhance the practicality, effectiveness, and adoption potential of RL-based local motion planners. We aim to address the reality gap and improve human-centered design for these systems.

Reality Gap. The reality gap presents a significant challenge in both online (simulation-based) and offline RL settings for our local motion planning tasks. Our approach focuses on bridging this gap, offering solutions applicable to a broad spectrum of data-driven robotic applications beyond just local motion planning for mobile robots. For online RL trained in simulation, we propose designing a method to incorporate dynamics mismatch information using a small dataset collected from the target environment. This approach should enable the policy to be aware of target dynamics during simulation-based learning, leading to improved real-world performance. We will explore techniques such as adaptive sim-to-real transfer, and uncertainty prediction to achieve this goal. In the offline RL context, we aim to bridge the reality gap that

may exist when using pre-collected data. This approach opens possibilities for leveraging more accessible source datasets, even if they do not have exactly similar dynamics to the target environment. This strategy could address the challenge of data scarcity in offline RL for robotics applications. To address these challenges, we propose exploring diffusion-based solutions, a novel approach that has shown great success in offline RL tasks and offers promising potential for tackling dynamics mismatch.

Human-centered Design. Our primary objective in handling human pedestrian behavior is to develop a robust agent capable of navigating the unpredictable nature of crowds. To improve the robustness of the RL agent, our key idea is to encourage diversity during the learning process with multiple agents. This diversity should be free from expert knowledge and minimize inductive bias. By exposing the agents to a wide range of experiences through interactions with diverse behaviors, we expect each agent to develop more robust and adaptable navigation strategies. This approach aims to create an RL-based motion planner that can navigate complex environments with unpredictable pedestrians, including in unseen scenarios. We also plan on expanding and refining the current testing framework for local motion planning among moving obstacles by developing a more comprehensive and diverse set of testing scenarios that better represent unpredictable human movement. These scenarios will be distinct from the training scenarios to ensure the agent's ability to generalize to unseen situations.

Finally, to enhance user experience and build trust in mobile robots, we propose personalizing the robot's behavior. Our research will focus on developing a resource-efficient method for rapid and accurate adaptation to individual user preferences across a wide range of automated decision-making systems. We will develop a practical framework based on a two-stage process: pretraining and fine-tuning. First, we will train a general policy that captures common behaviors and navigation strategies. Then, we will implement a quick fine-tuning mechanism that can adjust this general policy using just a few human preference labels. These preference labels will directly reflect the ideal behavior that each user wants to experience from the robot. This approach eliminates the need for hand-crafted reward shaping, which may difficult to accurately represent individual user preferences. Additionally, we will explore diffusion-based solutions, which have shown promising flexibility in several offline RL formulations, to guide the personalization process.

1.4 Main Contributions

The main contributions of each chapter are summarized below.

- **Mitigating the dynamics mismatch inherent in the sim-to-real gap when utilizing simulators for online RL**. We propose a novel approach using a single policy to predict two sets of actions: one for maximizing task rewards in simulation and another specifically for domain adaptation via reward adjustments. This decoupling makes it easier to maximize the overall reward in the source domain during training. Our proposed method is effective in bridging the sim-to-real gap, outperforming several baselines on challenging tasks. Additionally, by incorporating uncertainty-based exploration during training, our approach is shown to enhance agent robustness.

- **Mitigating the dynamics mismatch in Offline RL, and alleviating the data scarcity by enabling the use of more readily available, off-dynamics datasets**. We propose a novel approach using conditional Diffusion models to learn the joint distribution of the large-scale off-dynamics dataset and the limited target dataset. We introduce two contexts for the conditional model to capture the underlying dynamics structure: (1) a *continuous dynamics score* allows for partial overlap between trajectories from both datasets, providing the model with richer information; (2) an *inverse-dynamics context* guides the model to generate trajectories that adhere to the target environment's dynamic constraints. Empirical results show our method significantly outperforms several strong baselines. Ablation studies reveal the importance of each dynamics context. Additionally, the model's robustness to subtle environmental shifts is demonstrated through context modification.

- **Personalizing automated decision-making systems by introducing a resource-efficient approach that enables rapid adaptation to individual user's preferences**. We propose a novel approach to pretrain a conditional diffusion model on reward-free offline datasets. We introduce the concept of preference latent embedding and provide the detailed process of co-training it during the pretraining phase. Furthermore, we present an adaptation method for rapid preference alignment through preference inversion to

quickly fine-tune the pretrained model to user-specific preferences. We create a benchmark experiment using real human preferences on diverse, optimal trajectories, better reflecting real-world scenarios. Rigorous evaluations and ablation studies are conducted using both our human-annotated dataset and an existing dataset. It is shown that our proposed method demonstrates superior adaptation accuracy to human preferences with less data in both simulated and real-world scenarios. The ablation studies reveal that the best performing configuration utilizes spectral normalization for pretraining, followed by initializing the preference latent embedding using a uniform prior during adaptation.

- **Developing an RL-based local motion planner that can robustly navigate in environments with crowded and unpredictable human pedestrians**. We propose an efficient method that enhances the agent diversity within a single policy by maximizing an information-theoretic objective. This diversity enriches each agent's experiences, improving its adaptability to unseen crowd behaviors. A novel set of diverse scenarios inspired by pedestrian crowd behaviors are proposed to assess an agent's robustness against pedestrian behavior not encountered during training. Experiments shown that the proposed behavior-conditioned policies outperform existing works in these challenging scenes, reducing potential collisions without additional time or travel. Additional experiments validated that the methods remains effective in high density environments and more realistic evaluations in Gazebo simulator.

1.5 Thesis Organization

The remainder of this thesis is structured as follows: Chapter 2 introduces the fundamental concepts of RL, encompassing both online and offline approaches. It also explores RL methods in the context of local motion planning and presents the basics of Diffusion models, including their application to offline RL. Chapters 3 and 4 propose methods to address dynamics mismatch in online (simulation-based) and offline RL, respectively. The latter part of the thesis focuses on human-centered solutions for RL-based planners: Chapter 5 proposes an effective and efficient personalization framework using diffusion models. Chapter 6 tackles the challenge of navigation amidst crowded and unpredictable human pedestrians. Finally, Chapter 7 concludes the thesis, offering insights into relevant future research directions. Figure 1.1 provides a visual representation of our research and its chapter structure.

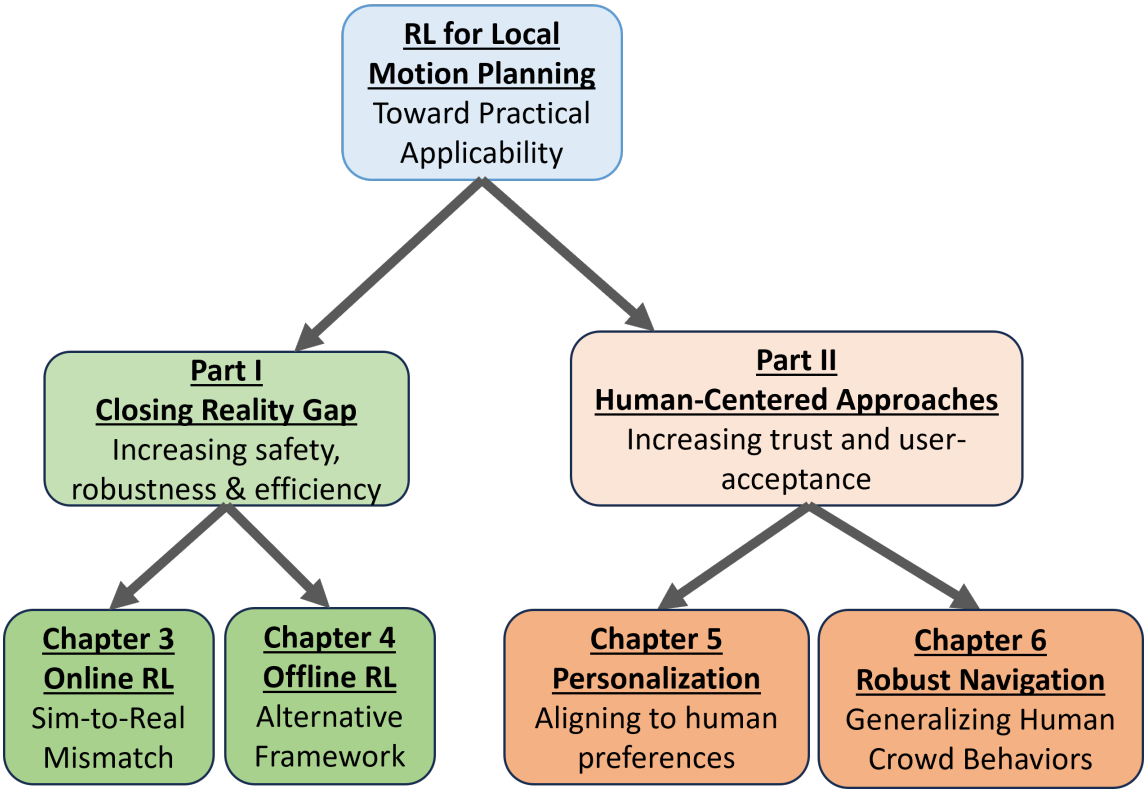


Figure 1.1: An overview of the research and chapter structure of this thesis.

Chapter 2

Preliminary

This chapter covers the core principles of deep reinforcement learning (RL). We begin with an overview of the Markov decision process, followed by a definition of RL and Offline RL, as well as a discussion of key RL algorithms. We then examine the RL framework as it applies to local motion planning. This discussion lays the groundwork for the concepts explored in Chapter 6. The chapter concludes with an introduction to Diffusion models and their use in offline RL. This modeling approach is central to the methods we propose in Chapters 4 and 5.

2.1 Reinforcement Learning

2.1.1 Markov Decision Process

Markov Decision Process (MDP) is a mathematical framework used to represent an environment in RL [38]. It provides a structured way to formulate the problem of learning from interactions to achieve a desired goal. Formally, a finite-horizon MDP can be represented by $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$:

- $\mathcal{S}$ - a set of states, $\mathcal{S} \subseteq \mathbb{R}^n$, which may be either discrete or continuous.
- $\mathcal{A}$ - a set of actions, $\mathcal{A} \subseteq \mathbb{R}^m$, which similarly can be discrete or continuous.
- The function $P(s'|s,a)$ represents the transition dynamics, indicating the likelihood of the environment moving from state s to state s' when action a is executed. Here, $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$.

- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines the reward mechanism. It assigns a real-valued reward to each state-action pair, where states are drawn from $\mathcal{S}$ and actions from $\mathcal{A}$.

- γ - discounting factor for future rewards, $\gamma \in (0, 1]$.

The MDP framework assumes all states possess the *Markov* characteristic, meaning future state predictions rely solely on the current state, independent of historical information: $\mathbb{P}_a[s_{t+1}|s_t] = \mathbb{P}_a[s_{t+1}|s_1, \ldots, s_t]$. In the RL paradigm, an agent engages with its environment in a sequential manner, accumulating rewards. At each time step t, the agent receives a state s_t from the environment and selects an action a_t. Upon action execution, the environment yields an immediate reward r_t and transitions to state s_{t+1}. The agent's behavior is governed by a policy $\pi : \mathcal{S} \to \mathcal{A}$, which dictates action selection. For non-deterministic policies, $\pi(a_t|s_t)$ represents the likelihood of choosing action a_t given state s_t. The agent-environment interaction generates a series of state-action-reward tuples: $\{(s_0, a_0, r_0), (s_1, a_1, r_1), \ldots, (s_t, a_t, r_t), \ldots\}$. This sequence, termed the trajectory, includes rewards $r_t = \mathcal{R}(s_t, a_t)$ at each time step t. RL aims to determine an optimal policy π^* that maximizes the expected sum of future rewards:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_t \mathcal{R}(s_t, a_t)\right], \tag{2.1}$$

where actions a_t are drawn from the policy distribution $\pi(\cdot|s_t)$ conditioned on state s_t at time t.

2.1.2 Value Functions

Two important functions evaluate the agent's performance based on future rewards: the state-value function and the Q-function (state-action value function). The state-value function $V_\pi(s)$ defines the expected sum of future rewards when following policy π from state s:

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right]. \tag{2.2}$$

The *Bellman equation* for $V^{\pi}(s)$ is:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}r_{t} \mid s_{0}=s\right] \\ &= \mathbb{E}_{\pi}\left[r_{t}+\sum_{t=1}^{\infty}\gamma^{t}r_{t} \mid s_{0}=s\right] \\ &= \sum_{a}\pi(a|s)\sum_{s'}P(s'|s,a)\left[\mathcal{R}(s,a)+\gamma\mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}r_{t} \mid s_{0}=s'\right]\right] \\ &= \sum_{a}\pi(a|s)\sum_{s'}P(s'|s,a)\left[\mathcal{R}(s,a)+\gamma V^{\pi}(s')\right]. \end{aligned} \tag{2.3}$$

The Q-function $Q_{\pi}(s,a)$ defines the expected value given an action and a state under policy π:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}r_{t} \mid s_{0}=s, a_{0}=a\right]. \tag{2.4}$$

2.2 Algorithms

RL methods aim to learn a policy, denoted as π, that guides an agent to make optimal decisions in an environment. There are two primary categories of approaches to achieving this: value-based and policy-based, which are introduced in Section 2.2.1 and Section 2.2.2, respectively.

2.2.1 Value-based Approaches

Value-based approaches leverage the concept of an optimal Q-function, $Q^{*}(s,a)$. This function represents the maximum expected cumulative reward that can be obtained by taking action a in state s and subsequently following an optimal policy:

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a). \tag{2.5}$$

In these approaches, the optimal policy π^{*} is determined by greedily selecting the action associated with the highest Q-value in a given state s:

$$\pi^{*}(s) = \arg\max_{a} Q^{*}(s,a). \tag{2.6}$$

The Bellman equation provides a recursive relationship for calculating the optimal Q-function:

$$Q^{*}(s,a) = \mathbb{E}_{s'}\left[\mathcal{R}(s,a)+\gamma\max_{a'}Q^{*}(s',a')\right]. \tag{2.7}$$

One well-known value-based algorithm, Q-learning [14], iteratively updates the Q-function using temporal difference (TD) error. The update rule is as follows:

$$Q(s,a) = Q(s,a) + \alpha \left[\mathcal{R}(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right], \tag{2.8}$$

where α represents the learning rate. Over time, this update process leads to the convergence of the estimated Q-function, $Q(s,a)$, to the optimal Q-function, $Q^*(s,a)$.

The Deep Q-Network (DQN) [39] is a notable advancement in value-based methods. It approximates the Q-function using a neural network with parameters θ, denoted as $Q(s,a;\theta)$. The training of this network involves minimizing a regression loss:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{s,a} \left[\left(r + \gamma \max_{a'} Q(s', \mathbf{a}'; \theta_{i-1}) - Q(s,a;\theta_i) \right)^2 \right], \tag{2.9}$$

where θ_i represents the parameters at iteration i. DQN utilizes a replay buffer to store and randomly sample past experiences during training, effectively mitigating the issue of correlated data. Each time step t results in a transition $(s_t,\ a_t,\ s_{t+1},\ r_t)$ being stored in the buffer. This buffer maintains a collection of recent transitions, discarding the oldest ones as it fills up. This strategy facilitates the neural network's update process by providing a diverse set of experiences.

2.2.2 Policy-based Approaches

In reinforcement learning, policy-based methods focus on directly learning the policy π, often represented by a neural network in the context of deep RL. Unlike value-based approaches, policy-based methods do not rely on explicitly estimating value functions.

Policy Gradient techniques, as described in Sutton et al. [40], model the action distribution of the policy and directly approximate the gradients of expected rewards with respect to the policy parameters. If we denote the policy parameterized by θ as $\pi_\theta(a|s)$ and the expected accumulated rewards as $J(\theta)$, the policy gradient can be estimated as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[Q^\pi(s,a) \nabla_\theta \log \pi_\theta(a|s) \right]. \tag{2.10}$$

Subsequently, the parameter θ is updated through gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta), \tag{2.11}$$

where $\alpha \in \mathbb{R}^+$ signifies the learning rate.

REINFORCE [41] estimates $Q^\pi(s,a)$ using Monte-Carlo (MC) methods:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[G_t \nabla_\theta \log \pi_\theta(a_t|s_t)\right], \tag{2.12}$$

where G_t is the accumulated discounted rewards from time step t.

Actor-Critic methods combine a policy model (actor) and a value function (critic). The critic evaluates the current policy by approximating the state-value, while the actor updates the policy parameters.

Advantage Actor-Critic (A2C) [42] uses the advantage value for policy updates:

$$A(s_t,a_t) = Q(s_t,a_t) - V(s_t) = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{2.13}$$

The critic is trained by minimizing the TD error $\delta_t = y_t - V_\theta(s_t)$, where $y_t = r_t + \gamma V(s_{t+1})$. The policy update becomes:

$$\theta \leftarrow \theta + \alpha A(s_t,a_t) \nabla_\theta \log \pi_\theta(a|s). \tag{2.14}$$

Proximal Policy Optimization (PPO) [43], a policy-based approach, introduces stability to the learning process by restricting the extent of policy updates. It achieves this by employing a clipped objective function:

$$\mathcal{L}^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(\dot{r}_t(\theta)\hat{A}_t, \text{clip}(\dot{r}_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]. \tag{2.15}$$

In this formulation, $\hat{\mathbb{E}}_t$ denotes the empirical expectation, $\hat{A}_t$ is the estimated advantage, ϵ is a hyperparameter controlling the clipping range, and $\dot{r}_t(\theta)$ represents the probability ratio between the updated and old policy:

$$\dot{r}_t(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}. \tag{2.16}$$

PPO's clipped objective function simplifies implementation and parameter tuning compared to related trust region methods like TRPO [44], while still delivering state-of-the-art performance.

Soft Actor-Critic (SAC) algorithm, introduced by Haarnoja et al. [45], is an actor-critic framework designed for learning stochastic policies in an off-policy fashion. While initially proposed for offline RL, where agents learn solely from a static dataset without direct environmental interaction, SAC has proven effective across various continuous control tasks in both offline and

online settings, making it a robust benchmark. In SAC, the stochastic policy is optimized using a critic network inspired by the Deep Deterministic Policy Gradient (DDPG) approach [46]. The critic network evaluates actions sampled from the policy and provides gradients for updating the actor network. A key aspect of SAC is its focus on maximizing policy entropy, which quantifies the randomness or exploratory behavior of the policy. The policy learning process in SAC balances exploitation (maximizing expected return) and exploration (maximizing policy entropy) through the objective function:

$$\mathcal{J} = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}\Big(\mathcal{R}(\mathbf{s}_t,\mathbf{a}_t)+\alpha\mathcal{H}\left(\pi(\cdot|\mathbf{s}_t)\right)\Big)\right], \tag{2.17}$$

where $\mathcal{H}$ represents entropy, and α is a hyperparameter that controls the trade-off between exploitation and exploration.

The Bellman equation for the entropy-regularized Q-function in SAC is given by:

$$\begin{aligned} Q^{\pi}(\mathbf{s},\mathbf{a}) &= \mathbb{E}_{\mathbf{a}'\sim\pi,\mathbf{s}'\sim P}\Big[\mathcal{R}(\mathbf{s},\mathbf{a})+\gamma\Big(Q^{\pi}(\mathbf{s}',\mathbf{a}')+\alpha\mathcal{H}\left(\pi(\cdot|\mathbf{s}')\right)\Big)\Big] \\ &= \mathbb{E}_{\mathbf{s}'\sim P}\Big[\mathcal{R}(\mathbf{s},\mathbf{a})+\gamma V^{\pi}(\mathbf{s}')\Big]. \end{aligned} \tag{2.18}$$

Similar to off-policy algorithms like DQN [39] and DDPG [46], SAC employs a parametric Q-function $Q(\mathbf{s},\mathbf{a})$ to approximate the on-policy Q-function $Q^{\pi}(\mathbf{s},\mathbf{a})$ and utilizes a replay buffer $\mathcal{D}$ to store experienced transitions. During learning, transitions are sampled from $\mathcal{D}$ to optimize the Q-function by minimizing the Bellman error:

$$\mathcal{L}(Q) = \mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{D}}\Big[\big(Q(\mathbf{s},\mathbf{a})-\mathcal{R}(\mathbf{s},\mathbf{a})-\gamma\bar{V}(\mathbf{s}')\big)^2\Big]. \tag{2.19}$$

Here, $\bar{V}$ is the target value network, which is updated to minimize the following mean squared error:

$$\mathcal{L}(\bar{V}) = \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\Big[\big(\bar{V}(\mathbf{s})-\mathbb{E}_{\mathbf{a}\sim\pi}\left[Q(\mathbf{s},\mathbf{a})-\alpha\log\pi(\mathbf{a}|\mathbf{s})\right]\big)^2\Big]. \tag{2.20}$$

The policy is evaluated using the parametric Q-function and improved by maximizing both the predicted Q-value and the entropy, with the objective:

$$\mathcal{J}(\pi) = \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\Big[\mathbb{E}_{\mathbf{a}\sim\pi}\left[Q(\mathbf{s},\mathbf{a})-\alpha\log\pi(\mathbf{a}|\mathbf{s})\right]\Big]. \tag{2.21}$$

Actions are sampled on-policy, and the reparameterization trick is employed to enable backpropagation through the policy network.

2.3 Offline RL

Offline RL is a data-driven approach to the RL problem where the objective remains to optimize the expected cumulative reward (Equation 2.1), but the agent cannot actively interact with the environment to collect new data. Instead, it relies on a fixed dataset of transitions, $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)\}$, analogous to a training set in supervised learning. This approach opens doors to training intelligent agents across a wide range of applications, including life-saving surgical robots [47,48], data-driven financial trading [49,50], improved medical diagnosis [51,52], and autonomous vehicles [53–55].

Offline RL presents unique challenges, notably the inability to gather new experiences through direct interaction with the environment. This limits discovery of high-reward regions if they are not represented in the static dataset $\mathcal{D}$. However, a more subtle yet crucial challenge is the need to address scenarios where the agent's actions differ from the observed data. This is necessary as the aim is to learn a policy that surpasses the demonstrated behavior in the dataset. This requirement leads to distributional shift, a common issue when a function approximator (e.g., policy, value function, or model) is trained on one distribution but evaluated on another. In offline RL, this shift stems from the divergence between the learned policy π_{off} and the behavior policy π_β used to collect the dataset. This divergence causes a change in the states visited and is further amplified by the act of maximizing expected returns.

Formally, the offline RL objective function $J(\phi)$ aims to minimize the Bellman error, calculated using the action-value Bellman equation:

$$J(\phi) = E_{s,a,s' \sim D}\left[\left(r(s,a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q_\phi(s',a')] - Q_\phi(s,a)\right)^2\right]. \tag{2.22}$$

This objective relies on the assumption that the action distributions of π_{off} and π_β are the same, which is often untrue in practice. Even with an accurate Q-function, errors can accumulate, leading to a divergence between the state distributions induced by the two policies: $d^{\pi_{\text{off}}}(s) \neq d^{\pi_\beta}(s)$. This divergence is more severe in offline RL as there is no ongoing interaction to rectify these errors.

2.3.1 Policy Constraints

A strategy to mitigate distributional shift in offline RL is to constrain the learning process, particularly by limiting how much the learned policy deviates from the behavior policy. This approach ensures that the Q-function is evaluated on actions that are within the distribution of the

training data, preventing the accumulation of errors due to out-of-distribution actions. Policy constraint methods for offline RL achieve this by enforcing a 'closeness' between the learned policy $\pi(a|s)$ and the behavior policy $\pi_\beta(a|s)$ using a probability metric. The specific metric and how the constraint is imposed can vary across different methods. One category involves explicit f-divergence constraints, where a direct constraint is added to the actor update to maintain proximity between the policies in terms of an f-divergence, typically the KL-divergence. Another category uses implicit f-divergence constraints, achieved through actor updates that inherently keep the policies close. The constraints can be implemented either directly on the policy update (policy constraints) or through a penalty added to the reward function or target Q-value (policy penalty). Formally, policy iteration with constraints can be represented as:

$$\hat{Q}^\pi_{k+1} \leftarrow \arg\min_Q \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}'\sim\pi_k(\mathbf{a}'|\mathbf{s}')}[\hat{Q}^\pi_k(\mathbf{s}',\mathbf{a}')]\right)\right)^2\right] \tag{2.23}$$

$$\pi_{k+1} \leftarrow \arg\max_\pi \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}[\hat{Q}^\pi_{k+1}(\mathbf{s},\mathbf{a})]\right] \text{ s.t. } D(\pi,\pi_\beta) \le \epsilon. \tag{2.24}$$

When these optimizations are not fully converged but limited to a few gradient steps, we get the actor-critic method with the additional constraint $D(\pi,\pi_\beta) \le \epsilon$ on the policy update. Prior methods have instantiated this approach with various choices of D [56–58], collectively referred to as policy constraint methods.

2.3.2 Conservation Value Functions

Instead of constraining the policy in an actor-critic framework, we can effectively regularize the value or Q-function to prevent overestimation of out-of-distribution (OOD) actions. This approach offers advantages like applicability to both actor-critic and Q-learning methods, even without an explicit policy representation, and eliminates the need for modeling the behavior policy. A simple modification to achieve a conservative Q-function is adding a penalty term to the objective function for fitting Q-function parameters. An example of this approach is Conservative Q-Learning (CQL) [59]. CQL modifies the standard Bellman error objective by introducing a conservative penalty term, resulting in a modified objective:

$$\tilde{J}(\phi) = J(\phi) + \alpha \cdot \text{penalty}(\phi), \tag{2.25}$$

where $\tilde{J}(\phi)$ is the Bellman error, penalty(ϕ) is the conservative penalty, and α is a weighting factor controlling the strength of the penalty. The penalty term encourages the Q-function to

underestimate the value of OOD actions, thereby promoting a more conservative policy that is less likely to take risky actions in states not well-represented in the dataset. One common form of the penalty is:

$$\text{penalty}_{\text{CQL}_0}(B, \phi) = \mathbb{E}_{s\sim B, a\sim\mu(a|s)}[Q_\phi(s, a)],$$

where $\mu(a|s)$ is an adversarial distribution chosen to maximize the Q-values. This effectively pushes down the Q-values for actions that are likely to be OOD. CQL has shown promising results in practice, achieving state-of-the-art performance on several offline RL benchmarks. Its simplicity and ease of implementation make it a popular choice for addressing the challenges of distributional shift and overestimation in offline RL.

2.3.3 Trajectory Optimization

Trajectory optimization in offline RL aims to learn a model of the trajectory distribution $P_{\pi_B}(\tau)$ induced by the behavior policy π_B [60]. This enables planning optimal actions from a given initial state s_0. Using a sequence modeling objective, anchored by multiple states and actions, reduces the risk of OOD actions and allows for utilizing large models like transformers [61].

Janner et al. [60] proposed the Trajectory Transformer (TT), which maximizes the log-likelihood of all tokens in a trajectory, represented as $\tau = (R_o, S_o, a_o, R_1, S_1, a_1, ...R_H, S_H)$ with horizon length H, and uses beam search for planning. Conversely, Chen et al. [15] introduced the Decision Transformer (DT), also based on the transformer architecture, but focusing on minimizing the mean squared error (MSE) between predicted and actual actions. They argue that predicting states and returns-to-go is not necessary for good performance. During evaluation, DT conditions trajectory rollouts on a target return and iteratively updates its reward-to-go target until episode termination. While computationally expensive, these transformer-based methods excel in sparse reward settings where traditional temporal-difference methods struggle due to their reliance on dense reward estimates for effective Q-value propagation.

2.4 RL For Local Motion Planning

In this section, we describe different approaches in which the local motion planning problem have been formulated into MDPs, $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$.

2.4.1 State Space

The state space in robotic motion planning typically consists of two distinct types: internal states and external states. This division allows for a comprehensive representation of both the robot's condition and its surrounding environment.

Internal States: Internal states capture the real-time information of the robot itself. In the pioneering work that proposed RL-based local motion planning, Chen et al. [15] utilized the robot's linear and angular velocities (v and ω) as key components of the internal state. Additionally, the desired goal position is represented by either the relative or absolute goal coordinates. Together, these elements constitute the internal states. This formulation has been widely adopted in subsequent research, with most follow-up work adhering to a similar structure for representing the robot's internal states.

External States: External states typically capture information about the robot's surroundings, providing crucial data for navigation and obstacle avoidance. Chen et al. [15] utilized ground truth data of obstacles as input. This approach implies that in real-world deployment, a front-end perception module would be required to process sensor data before feeding it to the RL module. Methods developed using ground truth data can establish an upper bound on performance since complete information about obstacles is available. However, this approach has drawbacks. The performance of the agent becomes highly dependent on the accuracy of the perception modules, and the policy might fail catastrophically given noisy inputs, as no noise is present during training. Furthermore, the policy might be overly optimistic due to the lack of obstacle occlusion in training scenarios. During training, the exact size, shape, and position of all obstacles are known, which may not always be observable in practice. An example of these approaches is illustrated in Figure 2.2.

Several research efforts have explored the use of end-to-end RL to directly predict actions from sensor data, eliminating the need for a separate front-end perception module. One of the pioneering works in this direction utilized RGB camera inputs for end-to-end RL in motion planning [62]. The primary advantage of RGB sensors is their ability to provide rich information, including the capability to classify different types of obstacles. However, a significant drawback is that training an end-to-end RL system with RGB inputs requires a highly realistic simulation of the real world to avoid distribution shift when deployed in real-world scenarios. This requirement can be computationally intensive and challenging to achieve.

Another frequently used sensor is the 2D Laser Scanner, initially proposed for the application in [17, 34]. The 2D laser scanner returns a one-dimensional array of distance measurements. Its main advantage lies in the relatively low distribution shift when transitioning from simulation to real-world deployment. Compared to RGB sensors, the simulation requirements for 2D laser scanners are less demanding in terms of world creation and simulation speed. Additionally, the distance readings from laser scanners are generally more reliable and offer shorter response time when dealing with moving obstacles.

In our work, we opt for a realistic representation that utilizes distance readings from a 2D laser range finder to sense the environment, similar to the approach in [17]. This choice strikes a balance between realism and practicality. We take into account the sensor noise and obstacle occlusions, making no assumptions about the shape, size, or number of obstacles. This approach aligns more closely with real-world conditions, potentially leading to more robust and generalizable motion planning strategies.

2.4.2 Action Space

The action space A represents the set of permissible velocities that a robot can execute, defined in either the continuous or discrete space. This space is crucial in determining the range and precision of movements the robot can perform during navigation and obstacle avoidance tasks. One commonly defined action space consists of two actions:

- Linear velocity: $v \in [0, v_{\max}]$. This represents the forward speed of the robot. The range starts from 0 (stationary) up to a maximum velocity $v_{\max}$. $v_{\max}$ is determined by the physical limitations of the robot or safety considerations.

- Angular velocity: $\omega \in [-\omega_{\max}, \omega_{\max}]$. This represents the turning rate of the robot. The range includes both positive and negative values, allowing for clockwise and counter-clockwise rotations. $\omega_{\max}$ is the maximum turning rate, again determined by the robot's capabilities or safety limits.

In this formulation, both v and ω are continuous values, allowing for smooth and precise control of the robot's movement. This continuous action space provides a high degree of flexibility in

navigation, enabling the robot to make fine adjustments to its trajectory as needed. The output actions v and ω drive the motion of the robot using the following kinematics equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 \\ \sin\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{2.26}$$

where $\dot{x}$, $\dot{y}$ and $\dot{\varphi}$ are derivatives in the global coordinate frame as illustrated in Figure 2.1. The matrix multiplication in this equation accounts for the robot's current orientation, φ, when calculating how the velocities affect its position and heading in the global coordinate frame. These equations describe how the linear and angular velocities translate into the robot's movement in the environment. Specifically:

- $\dot{x}$ represents the rate of change of the robot's x-coordinate.
- $\dot{y}$ represents the rate of change of the robot's y-coordinate.
- $\dot{\varphi}$ represents the rate of change of the robot's orientation angle.

Another common approach in defining the action space is to use a discretized version. This involves subdividing the continuous space into equal intervals or manually defining specific values. For example: linear velocity might be discretized into values like [0, 0.2, 0.4, 0.6, 0.8, 1.0] m/s and the angular velocity might be discretized into values like [-0.5, -0.25, 0, 0.25, 0.5] rad/s. Discretization can simplify the action selection process and may be beneficial in certain learning algorithms. However, it can also limit the precision of the robot's movements compared to a continuous action space. The choice between continuous and discrete action spaces often depends on factors such as the specific requirements of the task, the capabilities of the robot, and the complexity of the learning algorithm being used. Researchers and engineers must carefully consider these factors when designing the action space for their specific robotic application.

To execute these velocities on a wheeled robot, different drive controllers are typically used, depending on the robot's configuration and mechanical design. These controllers translate the desired linear and angular velocities into specific commands for the robot's motors or wheels. One commonly example is the differential drive controller. This is widely used for robots with two independently driven wheels. The controller calculates the required velocities for each

wheel based on the desired linear and angular velocities. The equations for left and right wheel velocities are:

$$\begin{aligned} v_l &= v - \frac{\omega L}{2} \\ v_r &= v + \frac{\omega L}{2}. \end{aligned} \tag{2.27}$$

where v_l and v_r are left and right wheel velocities, and L is the distance between the wheels.

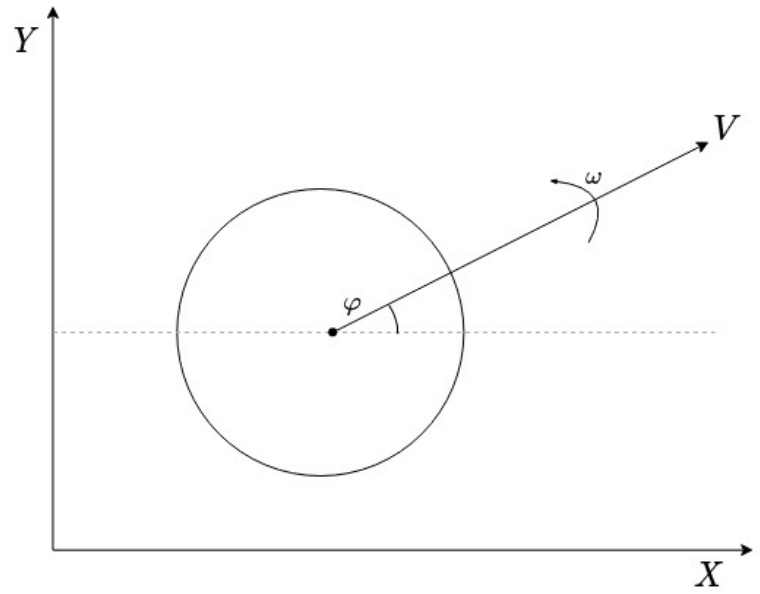


Figure 2.1: **A point robot in the global coordinate** (X,Y) **frame**. V and ω are the velocity and the angular velocity with respect to the robot's frame. φ is the angle between the two frames.

2.4.3 Rewards

To incentivize the robot to reach a goal without collision, some standard rewards commonly used in the literature are:

- Reaching Goal: it assigns a positive reward when the agent is within a certain distance of the desired goal. This reward encourages the robot to complete its primary objective.
- Crash: it assigns a negative reward when the robot is within a certain distance of any obstacles. This penalty helps the robot learn to avoid collisions.

- Timeout: it assigns a negative reward when the robot fails to reach the goal within a certain number of steps. This discourages inefficient or circuitous paths.

- Goal Approach Reward: it assigns a positive reward when the robot is closer to the goal than in the previous step. This provides a step-wise reward signal since the reach goal reward is very sparse. It helps guide the robot towards the goal even when it is far away.

Putting all together, the overall reward R can be represented as follows:

$$r_t = \begin{cases} r_{timeout} & \text{if } t < N_{\text{timeout}}, \\ r_{goal} & \text{if } \|\mathbf{p}_t - \mathbf{g}\| < d_{col}, \\ r_{col} & \text{else if collision }, \\ r_{step} \cdot (\|\mathbf{p}_{t-1} - \mathbf{g}\| - \|\mathbf{p}_t - \mathbf{g}\|) & \text{otherwise.} \end{cases} \tag{2.28}$$

where $r_{timeout} < 0$ is the penalty for not reaching the goal after N_{timeout} steps, r_{goal} is the reward for reaching the desired goal, $r_{col} < 0$ is the penalty for collision, r_{step} is the dense reward for getting closer to the goal, d_{col} is the distance threshold for reaching the goal, $\mathbf{p}$ and $\mathbf{g}$ are positions of the robot and goal.

These standard rewards form the foundation of most RL approaches in robot navigation. They balance the primary objectives of reaching the goal and avoiding obstacles while encouraging efficient movement. In addition to these commonly used rewards, several other non-standard rewards have been proposed to improve the robot's behavior:

- Social Reward [19]: it adds some rules on social behaviors like overtaking, crossing, and passing. This reward helps the robot navigate in a more socially acceptable manner when interacting with humans or other robots.

- Curiosity Reward [20]: this is an intrinsic reward to encourage exploration. It motivates the robot to explore unfamiliar areas of its environment, potentially leading to better overall performance.

- Safety Zone Penalty [23]: it discourages obstacles in a predefined area near the robot. This creates a buffer zone around the robot, enhancing safety in dynamic environments.

- Crowd Density Penalty [63]: it discourages the robot from visiting crowded areas. This helps the robot navigate more efficiently in populated environments and reduce the risk of collisions or social discomfort.

- Trajectory Smoothness [17]: it discourages the robot from taking large angular velocity values. This smooths out the trajectories to some extent, resulting in more natural and efficient movement patterns.

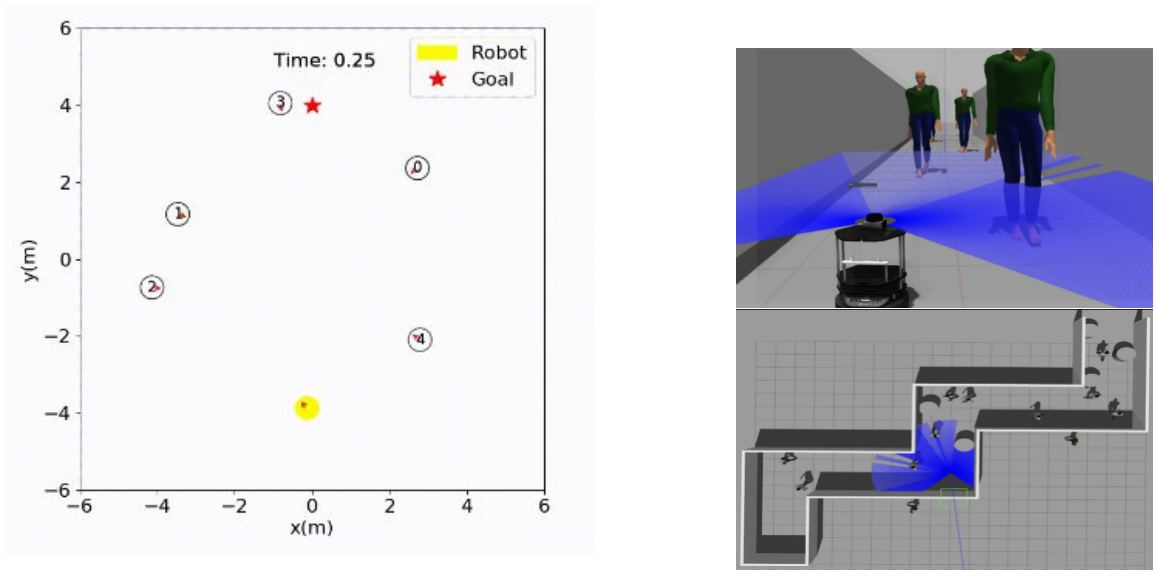
These additional rewards address specific aspects of robot behavior that go beyond the basic requirements of reaching a goal and avoiding obstacles. They can be combined with the standard rewards to create more sophisticated and context-aware navigation policies. During reward shaping, the relative weighting of each reward term can significantly impact the learned behavior. For example, placing too much emphasis on the crash penalty might result in overly cautious behavior, while overemphasizing the goal approach reward might lead to risky shortcuts. Moreover, the effectiveness of these rewards can vary depending on the specific environment and task. For instance, the social reward might be crucial in crowded human environments but less important in industrial settings.

2.4.4 Transition Probabilities

The transition probabilities are determined by the environment the RL agent interacts with. For our local motion planning task, two key factors affect the environment: physical properties of the robots and nature of the obstacles. For the environment, there are many proposed methods which train the agent in custom built simulators [16, 18–20] or Stage simulator [17, 21, 22]. These simulators follow the kinematics constraints described in Section 2.4.2. However, some physical properties like inertia, friction, stiffness, dampening and friction values which affect the dynamics of the robots are not considered. Some works [23–25] used the Gazebo simulator [64] to include such physical properties when training the robot.

Another key aspect that affects the transition probability is how the dynamic obstacles are generated. When our robot agent interacts with the environments, the movements of the dynamic obstacles will determine how the robot agent should move to avoid them. Fan et al. [21] design custom movements using waypoints for the obstacles. Chen et al. [18] utilize Optimal Reciprocal Collision Avoidance (ORCA) [65], an obstacle avoidance algorithm to control the dynamic obstacles. Sun et al. [26] simulate crowds based on social forces models.

The most popular way, however, is not to self define crowd movements. Instead, some works [15–18] proposed training in a new paradigm using a decentralised multi-agent scheme.



(a) Robot in custom 2D simulator using Ground Truth Position and Velocities of obstacles

(b) Robot in Gazebo Simulator using 2D laser Scanner to detect obstacles

Figure 2.2: **Comparisons of Simulator and Input Type**

In this formulation, N agents following a single policy are randomly deployed to reach their respectively goals. The advantage of this approach is that the movements of the obstacles are constantly updated, unlike previous approaches where the policy might overfit to the fixed behavior of the pedestrians. One drawback is this formulation encourages cooperative behaviors and thus making it over-optimistic when deployed in the real-world.

2.4.5 Neural Network Architectures

The backbone design of the NN architectures depends on the input type. For ground truth position, Chen et al. [15] use a simple fully connected feed-forward network to aggregate features from other obstacles. One major drawback is that the number of agents needs to be fixed and this network is sensitive to the ordering of the agents. To overcome this, Everett et al. [16] propose using Long Short-Term Memory (LSTM) [66] to make the input length invariant to the number of agents. This approach is sensitive to the ordering of the agents. Chen et al. [18] propose a pooling with attention architecture to make the architecture invariant to the ordering of the agents. The experiments shown that it outperforms all the prior methods.

For 2D laser scanners, Tai et al. [34] propose using a simple fully connected feed-forward network for a 10-dimensional laser scan readings. Long et al. [17] propose to use multiple 1D-convolutions to process a 512-dimensional laser scan readings. The convolution layers enable higher dimensional inputs while maintaining a small network size. Lastly, Tan et al. [22] propose to first represent the 2D laser scan inputs in its Cartesian coordinate frame and subsequently apply 2D-Convolutions.

2.5 Diffusion Models

2.5.1 Diffusion Probabilistic Models

Diffusion Probabilistic Models (DPMs) [67–69] are a class of generative models that learn data distributions $q(x)$ through an inverse modeling process that effectively reverses the addition of noise. The forward noising process involves taking data points sampled from the true data distribution $p_{\text{data}}(x)$ and subjecting them to a sequence of noise additions. This generates a Markov chain $x_0 : K$, where each step x_k is obtained by adding Gaussian noise to the previous step x_{k-1} according to the formula $x_k \sim \mathcal{N}(\sqrt{\alpha_k}x_{k-1}, (1-\alpha_k)I)$. The parameters $\alpha_{0:K}$ represent a predefined noise schedule, and K denotes the total number of diffusion steps.

The reverse process, known as the variational process, aims to recover the original data x_0 from the final noisy sample x_K. This is achieved through a series of iterative denoising steps, where each step x_{k-1} is estimated from the previous noisy sample x_k using a parameterized model $p_\theta(x_{k-1}|x_k) = \mathcal{N}(x_{k-1}|\mu_\theta(x_k, k), (1-\alpha_k)I)$. The initial sample x_K is typically drawn from a standard normal distribution, and the denoising process continues until the original data x_0 is reconstructed. To optimize the parameters of the model, a surrogate loss function is employed [69]:

$$\mathcal{L}(\theta) = \mathbb{E}_{k\sim[1,K],x_0\sim q,\epsilon\sim\mathcal{N}(0,I)} \|\epsilon - \epsilon_\theta(x_k, k)\|^2 . \tag{2.29}$$

In this loss function, ϵ represents the true noise added at each step, while $\epsilon_\theta(x_k, k)$ is the noise predicted by the model. By minimizing this loss, the model learns to accurately estimate the noise at each step, enabling it to effectively reverse the diffusion process and generate high-quality samples. The mean of the reverse Gaussian distribution used in the denoising process is given by:

$$\mu_\theta(x_k, k) = \frac{x_k - \sqrt{1-\bar{\alpha}}\epsilon_\theta(x_k, k)}{\sqrt{\bar{\alpha}}}, \tag{2.30}$$

where $\bar{\alpha}_k = \prod_{s=1}^{k} \alpha_s$ represents the cumulative product of the noise schedule parameters in up to step k. This formula ensures that the denoised samples gradually converge towards the original data distribution as the noise is progressively removed.

These generative models have garnered significant attention in recent years due to their ability to generate high-quality, diverse samples across various domains, including images, audios, and videos [70–75]. These generative models learn to construct data by reversing a process that systematically introduces noise into the data over multiple steps. This approach offers several key advantages. Firstly, it allows for the creation of flexible models that can accommodate arbitrary data structures, making them applicable to a wide range of tasks. Secondly, it enables training in a tractable manner, avoiding the instability issues often encountered in other generative models like Generative Adversarial Networks (GANs) [76, 77]. DPMs exhibit high stability during training and are less prone to mode collapse, a phenomenon where the model produces limited and repetitive samples.

2.5.2 Conditional DPMs

To enhance the capabilities of DPMs, researchers have introduced conditional DPMs through the classifier-guided sampling [71] or classifier-free [78] approach. These conditional models have been applied to a wide range of tasks across various domains. A notable application involves generating images from textual descriptions [70–72], where the generative model produces visual content based on the provided text input. This approach has shown remarkable results in creating images that closely match the given textual descriptions. In addition to utilizing textual contexts for conditioning, researchers have explored other types of contexts for conditional image generation. For example, low resolution contexts have been used for super-resolution tasks [79], binary masks contexts for in-painting applications [80], and semantic masks and human poses contexts for ControlNet [81]. These diverse conditioning approaches demonstrate the flexibility and adaptability of conditional DPMs in addressing various image generation challenges. Beyond image generation, conditional DPMs have also achieved significant success in various other fields. For instance, they have been applied to 3D graph generation using 2D molecular graph contexts [82], point-cloud synthesis from partial point-cloud observations [83], and audio generation conditioned on linguistic features [73]. These

applications highlight the versatility of conditional DPMs across different data modalities and problem domains.

Classifier-guided diffusion. To explicitly integrate class information into the diffusion process, Dhariwal and Nichol [71] proposed training a classifier $f_\phi(y|x_t,t)$ on noisy images x_t and using the gradients $\nabla_{x_t}\log f_\phi(y|x_t)$ to steer the diffusion sampling process towards the desired conditioning information y (such as a target class label). This is done by modifying the noise prediction. Recalling that $\nabla_{x_t}\log q(x_t) = -\frac{1}{\sqrt{1-\bar{\alpha}}}\epsilon_\theta(x_t,t)$, the score function for the joint distribution $q(x_t,y)$ can be approximated as:

$$\nabla_{x_t}\log q(x_t,y) \approx -\frac{1}{\sqrt{1-\bar{\alpha}}}\epsilon_\theta(x_t,t) + \nabla_{x_t}\log f_\phi(y|x_t). \tag{2.31}$$

This leads to a new classifier-guided predictor $\tilde{\epsilon}_\theta$:

$$\tilde{\epsilon}_\theta(x_t,t) = \epsilon_\theta(x_t,t) - \sqrt{1-\bar{\alpha}}\nabla_{x_t}\log f_\phi(y|x_t). \tag{2.32}$$

A weight w can be added to control the strength of the classifier guidance:

$$\tilde{\epsilon}_\theta(x_t,t) = \epsilon_\theta(x_t,t) - \sqrt{1-\bar{\alpha}}w\nabla_{x_t}\log f_\phi(y|x_t). \tag{2.33}$$

Classifier-free guidance. This method enables conditional diffusion steps without requiring a separate classifier [78]. This is achieved by combining scores from both conditional and unconditional diffusion models. Let $p_\theta(x)$ be an unconditional denoising diffusion model parameterized by a score estimator $\epsilon_\theta(x,t)$, and $p_\theta(x|y)$ be the corresponding conditional model parameterized by $\epsilon_\theta(x_t,t,y)$. These two models can be learned using a single neural network, where a conditional diffusion model is trained on paired data (x,y) with the conditioning information y occasionally discarded to enable unconditional image generation. The gradient of an implicit classifier can be expressed using conditional and unconditional score estimators. Substituting this into the classifier-guided modified score eliminates the dependence on a separate classifier:

$$\begin{aligned}\nabla_{x_t}\log p(y|x_t) &= \nabla_{x_t}\log p(x_t|y) - \nabla_{x_t}\log p(x_t)\\ &= \frac{1}{\sqrt{1-\bar{\alpha}}}(\epsilon_\theta(x_t,t,y) - \epsilon_\theta(x_t,t)).\end{aligned} \tag{2.34}$$

Therefore, the modified score becomes:

$$\tilde{\epsilon}_\theta(x_t,t,y) = (w+1)\epsilon_\theta(x_t,t,y) - w\epsilon_\theta(x_t,t). \tag{2.35}$$

This approach allows for more controlled generation of samples based on specific input conditions c. During the sampling process, the predicted noise is adapted to a weighted combination of conditional and non-conditional sampling:

$$\hat{\epsilon}_\theta(x_t \mid c) = (1+w)\epsilon_\theta(x_t \mid c) - w\epsilon_\theta(x_t \mid \emptyset), \tag{2.36}$$

where $\emptyset$ represents the null context, and w is a parameter that regulates the trade-off between sample quality and diversity by balancing the conditioned and unconditioned models. This formulation allows for fine-tuning the generation process to achieve desired characteristics in the output. In practice, the unconditioned model is often obtained by applying dropout on the context embedding during training. This technique helps in learning a more robust model that can generate samples both with and without conditioning information. They demonstrated that classifier-free guidance can effectively balance Frechet Inception Distance (FID), a metric used to distinguish between synthetic and real images.

While it is possible to train a conditional model using classifier-guided sampling [71], the classifier-free guidance approach offers several advantages. These include improved control over the generation process and superior performance in various tasks [72, 84]. The classifier-free method has become increasingly popular due to its effectiveness and ease of implementation in conditional DPM frameworks.

2.5.3 Diffusion-based Planners

Diffusion-based planners, which leverage DPMs to generate trajectories, have recently emerged as a promising solution to the challenges faced by offline RL methods discussed in [37, 85]. This approach is an example of trajectory optimization solution discussed in Section 2.3.3. These planners have demonstrated superior performance compared to existing offline RL approaches, offering the ability to specify flexible constraints and compose multiple skills [84, 86, 87]. Their versatility has led to adaptations in various reinforcement learning problem settings and tasks, including Multi-agent RL, Meta-RL, Multi-task RL, and Safe RL [88–91], where diffusion models' strengths are utilized to address specific challenges.

More formally, diffusion-based planners utilize expressive PDMs to model trajectories in the following form:

$$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{s}_0 & \boldsymbol{s}_1 & \dots & \boldsymbol{s}_H \\ \boldsymbol{a}_0 & \boldsymbol{a}_1 & \dots & \boldsymbol{a}_H \end{bmatrix}, \tag{2.37}$$

where H represents the planning horizon. These planners learn the gradient $\epsilon_\theta(\tau^i, i)$ associated with the denoising of trajectories. The mean μ_θ is computed in closed-form from the learned gradient, following the denoising framework of Ho et al. [69] (Equation 2.29). The model is trained using the simplified objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{i,\epsilon,\tau^0}\left[\|\epsilon - \epsilon_\theta(\tau^i, i)\|^2\right],$$

where:

- $i \sim \mathcal{U}\{1, 2, \ldots, N\}$ is a uniformly sampled diffusion timestep,
- $\epsilon \sim \mathcal{N}(0, I)$ is a Gaussian noise target, modeled by U-Nets [92], chosen for their non-autoregressive, temporally local, and equivariant characteristics
- τ^i represents the trajectory τ^0 perturbed by noise ϵ.

The covariance matrices Σ^i governing the reverse denoising process utilize the cosine-based scheduling proposed by [93]. While this process enables learning the underlying distribution of an offline dataset, naive sampling from the diffusion model yields trajectories that are randomly drawn from the learned distribution and do not necessarily maximise task-specific returns.

Diffuser. To generate trajectories that maximize the return, Diffuser [86] uses guided-diffusion with the gradient of the return:

$$\nabla \mathcal{J} = \sum_{t=0}^{T} \nabla_{s_t, a_t} r\left(s_t, a_t\right) = \nabla_{\boldsymbol{\tau}} \log p\left(O_{1:T} \mid \boldsymbol{\tau}\right), \tag{2.38}$$

where O_t is a binary random variable indicating the optimality of timestep t in a trajectory, and $p\left(O_t = 1\right) = \exp\left(\gamma^t r\left(s_t, a_t\right)\right)$. A separate model $\mathcal{J}_\phi$ is trained to predict the cumulative rewards. The gradients of $\mathcal{J}_\phi$ are then used to guide the trajectory following the classifier-guided sampling procedure.

Decision-Diffuser. Following the development of Diffuser, Decision-Diffuser [84] was introduced. This approach adopts a classifier-free method, with the optimization objective formulated as:

$$\max_\theta \mathbb{E}_{\tau \sim D}[\log p_\theta(x_0(\tau) | c(\tau))], \tag{2.39}$$

In this equation, $c(\tau)$ may encompass various conditional information related to the trajectory τ, such as the cumulative return or specific skills demonstrated within the trajectory. Decision-Diffuser demonstrates superior performance compared to Diffuser, with the added flexibility of incorporating new conditions. These conditions can significantly enhance the adaptability and robustness of the planner across various tasks. Examples of useful conditions include task abstractions, which provide high-level instructions or sub-goals to structure the trajectory generation process [90]; goal states, enabling trajectory optimization toward specific endpoints [86]; skills, which represent reusable motion primitives or behaviors that facilitate solving complex tasks [84]; and obstacle maps, which incorporate spatial information about the environment to ensure collision-free trajectory generation [94]. By leveraging these conditions, diffusion-based planners can tailor trajectories to meet specific requirements, improve task performance, and generalize across diverse scenarios.

Connection to Motion Planning Both Diffusion-based planners like Diffuser and Decision-Diffuser enable an agent to generate high-reward trajectories from any given state. In practice, while these methods sample complete trajectories, only the first action of the sampled trajectory is executed in the environment at each step. The remaining actions in the trajectory are discarded but play a critical role during the planning process. Specifically, adjacent actions within the sampled trajectory ensure local consistency, meaning that consecutive actions align smoothly without abrupt transitions. Additionally, the long planning horizon of the trajectory guarantees global coherence, where the overall trajectory adheres to the desired objective or task. This balance between local consistency and global coherence is crucial for effective motion planning, as it ensures that the agent's immediate actions contribute to achieving long-term goals while maintaining smooth and stable operation.

Part I

Closing the Reality Gap: Online & Offline Strategies

Chapter 3

Mitigating Dynamics Mismatch in Sim-to-Real Reinforcement Learning

3.1 Introduction

RL is a powerful technique for training intelligent agents to make sequential decisions. The online nature of this approach requires agents to interact directly with the deployed environment, iteratively collecting experience and improving their decision-making capabilities in real-time. This direct interaction with the real world, while potentially beneficial for learning, poses significant challenges in terms of safety, cost and time efficiency. Particularly, training RL agents directly in the real world presents numerous challenges. Real-world interactions can be expensive, time-consuming, and even dangerous, especially when dealing with complex systems like robots or autonomous vehicles [95, 96].

Simulators offer a compelling solution to the challenges of training online RL agents in the real world [97, 98]. They provide a safe, controlled, and cost-effective environment where agents can learn through trial and error, without the risks and costs associated with real-world interactions. In a simulator, agents can interact with virtual replicas of the real world, allowing them to freely explore different strategies, make mistakes, and learn from them without any negative consequences. Simulators often run faster than real-time, enabling rapid iteration and experimentation, and they allow for parallel training of multiple agents, significantly accelerating the learning process. This accelerated learning is particularly valuable for developing complex behaviors that would be impractical or even dangerous to learn directly in the real world. Furthermore, simulators offer researchers a high degree of flexibility in designing scenarios, enabling them to expose agents to a wide range of situations and edge cases, thereby

enhancing their robustness and ability to generalize to new situations. This shift from the real world to simulated environments opens the door to training intelligent agents for various applications, paving the way for advancements in fields like robotics [97, 98], healthcare [99], and finance [100].

However, the use of simulators in online RL introduces the challenge of the *sim-to-real gap* between the simulated and real-world environments [27,29]. A major contributor to the sim-to-real gap is the *dynamics mismatch*, where the physical dynamics of the simulated environment differ from those of the real world. This mismatch can manifest in various ways across different domains. In robotic manipulation, simulators often struggle to accurately model friction and contact dynamics, leading to discrepancies in grasping and object interaction behaviors [27]. For autonomous vehicles, the complex tire-road interactions and aerodynamic effects at high speeds may not be fully captured. In aerial robotics, the impacts of air turbulence and intricate propeller dynamics are frequently oversimplified [101]. Legged robots face challenges with accurate terrain interaction modeling, particularly on deformable surfaces [97, 102]. Underwater and soft robotics encounter issues with fluid dynamics and material deformation simulations [103]. These discrepancies in dynamics can lead to learned policies that fail to generalize or perform sub-optimally when transferred to the real world, as the agent's expectations about the consequences of its actions may not align with reality. Despite efforts to create realistic simulators, discrepancies between simulated and real-world environments can lead to performance degradation when deploying agents trained solely in simulation to the physical world.

Several approaches have been explored to tackle the sim-to-real gap, and this work specifically focuses on the challenge of *dynamics mismatch*. One common approach is domain randomization (DR) where RL policies are trained across a diverse range of simulated dynamics [27, 28]. While DR improves real-world performance by training adaptable policies, it often requires prior knowledge of parameter variations. Another contrasting approach involves grounding the simulator to resemble the target domain, allowing the agent to learn as if it were directly interacting with the real world [104–106]. A similar strategy involves reward shaping, where a bonus reward incentivizes the agent to take actions that mimic those preferred in the real world [107, 108]. Both grounding and reward shaping techniques require some level of interaction with the target environment, which contradicts our objective of avoiding real-world interactions due to safety and feasibility concerns. Therefore, we focus on a more realistic

approach by leveraging a small, sub-optimal offline dataset from the target environment to reduce the sim-to-real gap [109]. This eliminates the need for further real-world interactions, making it well-suited for real-world scenarios, e.g., operating rescue robots in hazardous environments, deploying trading strategies in high-frequency markets, or controlling unmanned aerial vehicles in challenging settings.

In this work, we propose a novel method, *Dual Action Policy* (DAP), which utilizes a single policy to simultaneously predict two distinct sets of actions. The first set of actions functions conventionally, maximizing the task reward within the simulation environment. However, a crucial addition is the introduction of a second set of actions which enables the policy to separately address the dynamics mismatch. Inspired by [107], we achieve this by incorporating a reward adjustment which incentivizes the policy to prioritize actions that lead to state transitions resembling those in the target domain. By decoupling the actions, the agent can more easily maximize the task reward while also attending to the reward adjustments. Additionally, to increase the robustness of our agent, we propose an exploration method based on the inherent uncertainty in dynamics estimation. This encourages the agent to actively explore areas with high uncertainty during training, enabling it to remain within the state-action distribution of the data during deployment.

Our experiments demonstrate the effectiveness of DAP for bridging the sim-to-real gap. We compared various methods on challenging simulated tasks with mismatched dynamics. DAP outperformed all baselines, achieving significantly higher returns than the strong baselines. Additionally, incorporating uncertainty estimation further improved performance, nearly matching the optimal results in some cases. Ablation studies validated the importance of the uncertainty-based exploration and showed that DAP remains effective even with a limited amount of target data.

3.2 Related Work

3.2.1 Dynamics Mismatch in Sim-to-Real RL

The challenge of bridging the sim-to-real gap in RL, particularly the issue of dynamics mismatch between training and deployment environments, has spurred the development of various methodologies. These methods can be broadly classified into three main approaches: system

identification, domain randomization, and domain adaptation. Each approach offers unique advantages and limitations in tackling the complexities of transferring knowledge learned in simulation to real-world scenarios.

System identification, a well-established approach with a rich history, involves utilizing offline data collected from the target environment to calibrate the simulator's parameters. By adjusting these parameters, the simulated environment can be made to more closely resemble the real world, enhancing the transferability of learned policies [29]. An extension of this is online system identification, where inferred system parameters are directly utilized during the training process to update a meta-policy in real time [28]. This allows for continuous adaptation and refinement of the policy as the agent interacts with the real-world environment. While effective, both offline and online system identification methods can be data-intensive and computationally demanding, particularly in complex, high-dimensional environments.

Domain randomization (DR) presents a contrasting approach to addressing the sim-to-real gap by training RL policies across a wide range of simulated dynamics. This technique involves systematically varying the parameters of the simulated environment during training, exposing the agent to a diverse set of conditions and scenarios. By doing so, DR encourages the learning of robust policies that can generalize to a variety of real-world conditions [27]. The randomized parameters can include physical properties such as mass, friction coefficients, and object dimensions, encouraging the agent to develop meta-capabilities or learn a meta-policy that generalizes across diverse dynamics. This approach emphasizes adaptability and robustness, aiming to create policies that are less sensitive to the specific dynamics of any single environment. Instead, the agent learns to perform well across a distribution of possible dynamics, increasing the likelihood of successful transfer to the real world. However, DR often requires careful consideration and prior knowledge or assumptions about the types and ranges of parameter variations that the agent may encounter in the real world. Determining the appropriate distribution of randomized parameters is crucial, as overly conservative randomization may lead to suboptimal performance, while excessive randomization can make the learning task too challenging [110–112].

Domain adaptation represents a more recent set of strategies aimed at bridging the sim-to-real gap. A prominent example is the reward adjustment technique, which penalizes actions during training that lead to transitions significantly different from those observed in the target domain [107]. This approach leverages auxiliary classifiers trained to distinguish between

source and target domain transitions, guiding the agent towards behaviors that are more aligned with the real-world deployment environment. The reward adjustment technique has demonstrated its versatility by being successfully applied to various RL domains, including model-based RL, inverse RL, and unsupervised RL, for mitigating dynamics mismatch [113–115]. Another approach in this category combines an offline dataset with online samples from simulation, offering a blend of offline and online learning, but it relies on the assumption of near-optimal samples in the offline dataset for effective performance [108].

Lastly, a hybrid category of approaches combines elements of both system identification and domain adaptation. These methods, such as grounded simulators, employ transformations on the source domain to align its behavior with the target domain, effectively allowing the agent to learn as if it were directly interacting with the real world [104–106]. This is achieved by incorporating forward and/or inverse dynamics models of both the source and target domains. While promising, the performance of grounded simulators is highly dependent on the state-action coverage achieved by the grounding policy, making it sensitive to the quality and diversity of the collected data.

3.2.2 Uncertainty in Deep Reinforcement Learning

Deep Neural Networks (DNNs) have been instrumental in the success of deep RL due to their ability to represent complex functions [39], including smooth dynamics often present in robotics [46]. However, when data is limited, DNNs are susceptible to over-fitting, which can lead to significant uncertainty about the model's true capabilities and degrade the performance of deep RL frameworks.

To address this issue, researchers have explored various approaches. One systematic method is parametric Bayesian inference [116], which leverages predefined probability distributions to represent uncertainty in model parameters. While powerful, this approach can be computationally expensive for complex models with numerous parameters, especially when dealing with high-dimensional data. Its effectiveness also heavily depends on choosing an informative prior.

An effective alternative is non-parametric bootstrapping, which utilizes an ensemble of models with random initializations. This technique has found wide application across various areas of deep RL. For example, in model-based RL, bootstrapping ensembles have been

employed to learn accurate dynamics models [60, 117]. They have also proven valuable in policy and value function estimation within model-free RL [118, 119] and offline RL [120, 121]. Furthermore, model uncertainty estimation plays a crucial role in designing exploration rewards [119, 122], enhancing the agent's ability to efficiently explore its environment. By incentivizing the agent to gather information that reduces model uncertainty, these approaches can lead to more efficient and effective learning, particularly in sparse reward settings. In our work, we leverage bootstrapped models to estimate uncertainty in the dynamics modelling, a factor we demonstrate to be crucial for domain adaptation.

3.3 Background

3.3.1 Problem Formulation: Off-Dynamics RL

We intend to deploy our agent in a given environment $\mathcal{M}_{\text{target}} = (S, A, P, R, \gamma, d_0)$. However, due to the constraints on interacting directly with the target environment, we leverage interactions with a more accessible simulation environment, $\mathcal{M}_{\text{source}} = (S, A, P', R, \gamma, d_0)$. The key difference between these two environments lies in their transition dynamics, i.e. $P \neq P'$. In our problem setting, we assume abundant interactions with $\mathcal{M}_{\text{source}}$ and a limited and sub-optimal dataset collected from $\mathcal{M}_{\text{target}}$.

3.3.2 Domain Adaptation with Rewards from Classifier (DARC)

To mitigate the dynamics mismatch, Eysenbach et al. [107] proposed a domain adaptation method DARC, which learns a policy whose behavior receives high reward in the source domain and has high likelihood under the target domain dynamics. By defining $p(\tau)$ as the desired distribution over trajectories in the target domain,

$$p(\tau) \propto p_1(s_1)\left(\prod_t p_{\text{target}}(s_{t+1} \mid s_t, a_t)\right)\exp\left(\sum_t r(s_t, a_t)\right), \tag{3.1}$$

and $q(\tau)$ as our agent's distribution over trajectories in the source domain,

$$q(\tau) = p_1(s_1)\prod_t p_{\text{source}}(s_{t+1} \mid s_t, a_t)\pi_\theta(a_t \mid s_t). \tag{3.2}$$

They minimized the reverse KL divergence between $p(\tau)$ and $q(\tau)$ as follows:

$$\min_{\pi(a|s)} D_{\mathrm{KL}}(q\|p) = -\mathbb{E}_{p_{\text{source}}}\left[\sum_t r\left(s_t, a_t\right) + \mathcal{H}_\pi\left[a_t \mid s_t\right] + \Delta r\left(s_t, a_t, s_{t+1}\right)\right] \tag{3.3}$$

where

$$\Delta r\left(s_t, a_t, s_{t+1}\right) \triangleq \log p\left(s_{t+1} \mid s_t, a_t\right) - \log q\left(s_{t+1} \mid s_t, a_t\right).$$

$\mathcal{H}_\pi$ is the entropy of the policy and t is the time-step. The reward adjustment Δr, penalises the agent for taking transitions more likely in the source domain than in the target domain and vice versa. The reward adjustment Δr requires an explicit model of the dynamics which may be inaccurate in continuous control tasks with a high dimensional state-action space. Subsequently, they estimated Δr using Bayes rules with two domain classifiers $p(\cdot|s_t, a_t, s_{t+1})$ and $p(\cdot|s_t, a_t)$ as follows:

$$\begin{aligned}\Delta r\left(s_t, a_t, s_{t+1}\right) = &\log p\left(\text{target} \mid s_t, a_t, s_{t+1}\right) - \log p\left(\text{target} \mid s_t, a_t\right) \\ &- \log p\left(\text{ source } \mid s_t, a_t, s_{t+1}\right) + \log p\left(\text{ source } \mid s_t, a_t\right)\end{aligned} \tag{3.4}$$

The domain classifiers are binary classifiers trained separately used to distinguish between source and target domain transitions.

3.4 Methodology

The key to DARC for domain adaptation lies in the introduced reward adjustment. This adjustment incentivizes agents to choose actions that lead to transitions resembling those in the target domain. However, this approach has two limitations:

1. **Action Set Constraint**: During training, the action set used to compute the reward adjustments (Equation 3.3) coincides with the action set used for updating the maximum entropy RL method through simulation sampling. Restricting both actions to the same set hinders the search for an optimal solution that maximises both rewards.

2. **Classifier Errors**: Due to the epistemic errors in the DARC classifiers, the dynamics reward adjustments might be inaccurate. These errors are particularly problematic if they are overly optimistic. In such cases, the policy might be led to sample state-action pairs that fall outside the target distribution during rollouts in the target environment. Consequently, the agent lacks the capability to self-correct and return to the desired distribution, causing planning in the target environment to diverge.

In the following, we propose novel solutions to address these limitations.

3.4.1 Dual Action Policy (DAP)

The first limitation of DARC lies in the action set constraint. To address this issue, we propose a relaxation strategy using DAP. The core idea of DAP is to utilize a single policy to simultaneously predict two distinct sets of actions $a = [a^{\text{src}}, a^{\text{tgt}}]$. The first set a^{src}, is used for sampling within the simulation environment which aligns with the standard behavior of maximum entropy RL methods. The second set introduces a novel concept: predicting an additional set of actions a^{src}. The decoupling of the actions into two sets would make it easier for a^{tgt} to address the dynamics mismatch via reward adjustments, while a^{src} focuses on maximizing the task reward. We highlighted all terms related to the a^{src} or a^{tgt} in blue and red respectively. Formally, following Equation 3.3, the modified objective function can be expressed as:

$$\min_{\pi([a_t^{\text{src}}, a_t^{\text{tgt}}]|s)} D_{\text{KL}}(q\|p) = -\mathbb{E}_{p_{\text{source}}}\left[\sum_t r\left(s_t, a_t^{\text{src}}\right) + \mathcal{H}_\pi\left(a_t^{\text{src}} \mid s_t\right) + \Delta r\left(s_t, a_t^{\text{tgt}}, s_{t+1}\right)\right]$$

and the modified reward adjustment is:

$$\hat{\Delta} r\left(s_t, a_t^{\text{tgt}}, s_{t+1}\right) \triangleq \log p\left(s_{t+1} \mid s_t, a_t^{\text{tgt}}\right) - \log q\left(s_{t+1} \mid s_t, a_t^{\text{tgt}}\right)$$

This objective is optimized under a new MDP, $\mathcal{M}_{\text{dual}} = (S, A_{\text{dual}}, P, R, \gamma, d_0)$, where $|A_{\text{dual}}| = 2 \times |A|$, and A is the original action-space. Similar to DARC, following Equation 3.4, we use a pair of domain classifiers to estimate $\hat{\Delta} r$:

$$\begin{aligned}\hat{\Delta} r\left(s_t, a_t^{\text{tgt}}, s_{t+1}\right) = &\log p\left(\text{target} \mid s_t, a_t^{\text{tgt}}, s_{t+1}\right) - \log p\left(\text{target} \mid s_t, a_t^{\text{tgt}}\right) \\ &- \log p\left(\text{ source } \mid s_t, a_t^{\text{tgt}}, s_{t+1}\right) + \log p\left(\text{ source } \mid s_t, a_t^{\text{tgt}}\right)\end{aligned} \tag{3.5}$$

An overview of the training process for DAP is illustrated in Figure 3.1. During deployment, we exclusively utilize a^{tgt} to sample actions in the target environment, while a^{src} is no longer used, as illustrated in Figure 3.2.

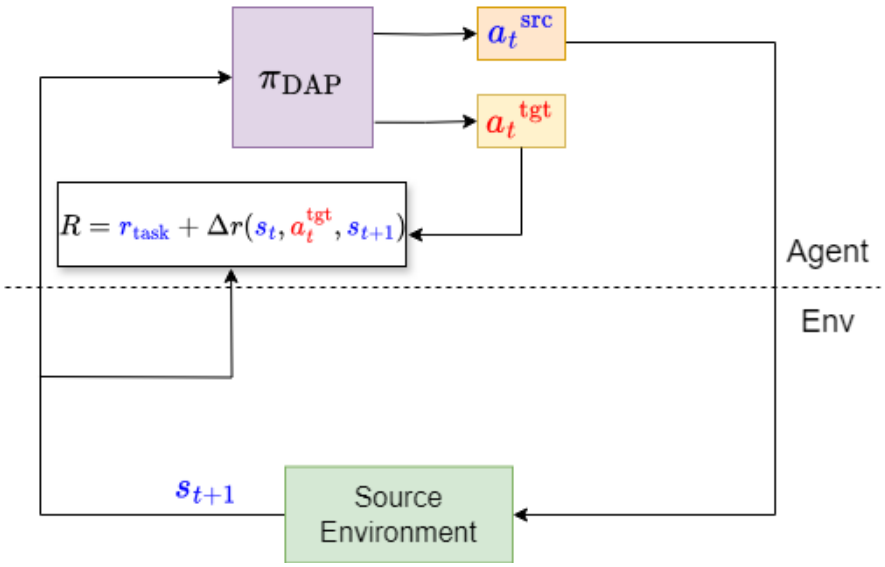


Figure 3.1: **Training in Source Env.** π_{DAP} predicts additional action set a^{tgt}, optimized for the target environment while a^{src} is utilized for sampling the source environment.

3.4.2 Regularization

From our DAP formulation in Equation 3.5, the source and target policies play distinct but interconnected roles. The source policy faces a dual optimization challenge: it must maximize returns while generating state sequences indistinguishable from the target tasks by the classifiers. In contrast, the target policy's primary influence is on the reward shaping term. This design creates an interesting dynamic. To this end, we introduce a regularization term, $||a_t^{\text{src}} - a_t^{\text{tgt}}||_2^2$, to prevent the generation of infeasible actions by a^{tgt}, controlled by a hyperparameter λ as follows:

$$\Delta r\left(s_t, a_t^{\text{tgt}}, s_{t+1}\right) = \hat{\Delta} r\left(s_t, a_t^{\text{tgt}}, s_{t+1}\right) + \lambda ||a_t^{\text{src}} - a_t^{\text{tgt}}||_2^2 \tag{3.6}$$

The regularization hyper-parameter λ plays a pivotal role in maintaining the balance between these two policies. Without regularization ($\lambda \to 0$), the target policy would be unconstrained and might prioritize generating actions that maximize the reward shaping term, without necessarily optimizing for actual task returns. Conversely, large λ values would cause DAP to converge towards DARC-like behavior, as the regularization term would dominate the reward shaping, effectively forcing the two policies to be nearly identical.

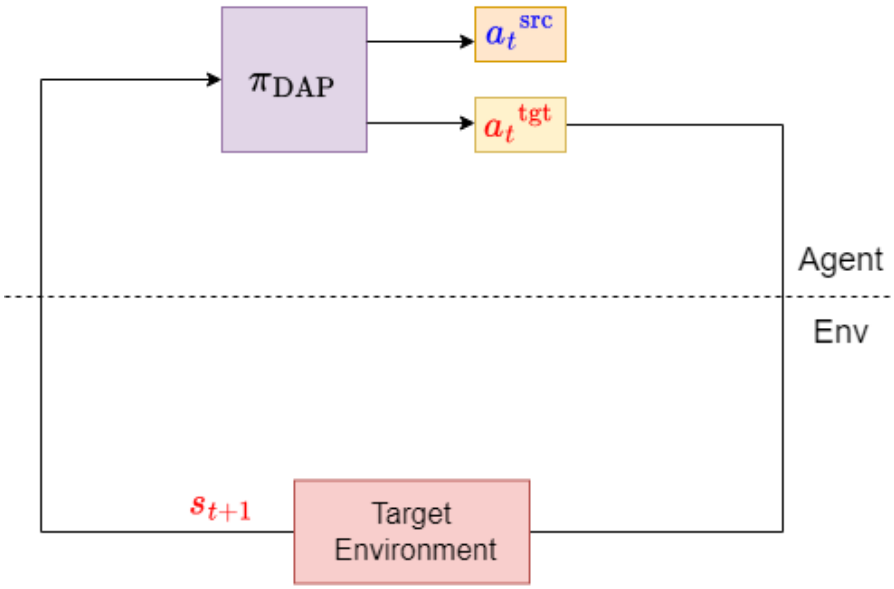


Figure 3.2: **Deployment in Target Env.** Only a^{tgt} is utilized while a^{src} is discarded.

Another factor contributing to DAP's effectiveness is the simultaneous prediction of a^{src} and a^{tgt} using a single policy. Maximizing Equation 3.6 with a single policy constrains the target policy to ensure that a^{tgt} stays within the state distribution reachable by a^{src}, which would be challenging if separate models were used.

3.4.3 Uncertainty-based Robust Action Resampling

Next, we address the problem of epistemic errors in the DARC's domain classifiers, which can lead to overly optimistic strategies during deployment. Our solution tackles this issue within the training framework by first measuring these uncertainties. Thereafter, we modify the predicted action to a more robust choice based on the severity of the uncertainty. This is achieved by randomly perturbing the action with a magnitude proportional to the uncertainty level. This approach allows actions with low uncertainty to remain unchanged, while forcing uncertain actions to explore a wider range of states. By encouraging exploration in uncertain areas, the agent gains the capability to self-correct and return to state-action distributions with greater certainty.

Formally, to quantify the dynamics uncertainty due to epistemic errors, we follow a simple but effective method in [123] which utilizes a deep ensemble. Specifically, we train an ensem-

ble of N domain classifiers (Equation 3.6), denoted by $p_i(\text{target}|s_t, a_t, s_{t+1})$ and $p_i(\text{source}|s_t, a_t, s_{t+1})$, respectively, for $i = 1, \ldots, N$, with randomly initialized weights. The intuition behind this approach is that a high standard deviation in the log probabilities across the ensemble indicates significant disagreement about the predicted state transitions, suggesting higher uncertainty in the dynamics model. During training, for each sampled action $a = [a^{\text{src}}, a^{\text{tgt}}]$, we resample and replace a^{src} with a robust action, denoted as $\hat{a}_t^{\text{src}}$, from a normal distribution:

$$\hat{a}_t^{\text{src}} \sim \mathcal{N}(a_t^{\text{src}}, k\sigma_t), \tag{3.7}$$

where

$$\begin{aligned}\sigma_t = \text{Std}\Big\{ \Big[&\log p_i\left(\text{target} \mid s_{t-1}, a_{t-1}^{\text{tgt}}, s_t\right) - \log p_i\left(\text{target} \mid s_{t-1}, a_{t-1}^{\text{tgt}}\right) \\ &- \log p_i\left(\text{source} \mid s_{t-1}, a_{t-1}^{\text{tgt}}, s_t\right) + \log p_i\left(\text{source} \mid s_{t-1}, a_{t-1}^{\text{tgt}}\right) \Big]_{i=1,\ldots,N} \Big\}\end{aligned} \tag{3.8}$$

Here, k represents a scaling hyper-parameter, σ_t represents an uncertainty measure and Std() is the standard deviation function. The modified action encourages exploration where the agent may behave erroneously in the target environment, enhancing its robustness. During deployment, the agent directly relies on the sampled action from the policy, bypassing the robust action. The complete training procedure combining DAP and uncertainty-based action resampling is detailed in Algorithm 1.

Algorithm 1 Dual Action Policies (DAP) with Uncertainty-based Action Resampling

Input: Target dataset $\mathcal{D}_{\text{target}}$, Source MDP $\mathcal{M}_{\text{source}}$
Input: Regularizer λ, scaling parameter k, ensemble size N.
Initialize: π_{DAP}, $\mathcal{M}_{\text{dual}}$ using $\mathcal{M}_{\text{source}}$, initial state s_1, replay buffer $\mathcal{D}_{\text{source}}$ and ensemble of domain classifiers $p_i(\cdot|s_t, a_t, s_{t+1})$ and $p_i(\cdot|s_t, a_t)$ for $i = 1, \cdots, N$.
for t in $1, 2, \ldots,$ num_iter **do**
    Sample $a = [a^{\text{src}}, a^{\text{tgt}}]$ from $\pi_{\text{DAP}}(s_t)$
    Resample a^{src} based on Eqn. 3.7 to get $\hat{a}^{\text{src}}$
    Sample s_{t+1} in MDP $\mathcal{M}_{\text{dual}}$ using $\hat{a}^{\text{src}}$
    Compute reward adjustment $\Delta r(s_t, a_t^{\text{tgt}}, s_{t+1})$ using Eqn 3.6
    Store $(s_t, a^{\text{src}}, a^{\text{tgt}}, r_t + \Delta r, s_{t+1})$ in $\mathcal{D}_{\text{source}}$
    Update π_{DAP} with $\mathcal{D}_{\text{source}}$ using SAC [45]
    Update domain classifiers with $\mathcal{D}_{\text{source}} \cup \mathcal{D}_{\text{target}}$ [107]
end for
return π_{DAP}

3.5 Experiments

In this section, we conduct experiments to demonstrate the effectiveness of our method. We begin by outlining the experimental setup, which involves creating a diverse and challenging set of environments for both training and evaluation. We then design and conduct comprehensive experiments to thoroughly compare the effectiveness of our proposed methods against several strong baselines. Additionally, we perform ablation studies to investigate the impact of regularizer, resampling the robust action and size of dataset.

3.5.1 Experimental Setup

Environments. To evaluate our method's ability to bridge the sim-to-real gap, we conducted experiments in the MuJoCo physics simulator [124]. We used a diverse set of four challenging settings created by modifying the physical properties of simulated robots in these environments: Ant, Half-Cheetah, Hopper, and Walker2d. We collected an offline dataset consisting of $M = 20000$ samples using a source behavioral policy sampled in the target environment. The source behavioral policy is trained using Soft-Actor Critic (SAC) in the source environment with 1M steps [45].

Baselines. We benchmarked against various sim-to-real algorithms which aim to address the dynamics mismatch. We have excluded methods which requires dynamics prior [27, 29, 109]. Overall, we trained the following policies:

1. **RL on source** - Policy trained in the source environment using SAC [45]. This is also the behavioral policy used to collect the target dataset.

2. **RL on target** - Policy trained in the target environment using SAC [45]. *Oracle* for the upper bound performance in the target environment.

3. **GARAT** [125] - Method to ground the simulator using adversarial policy.

4. **H2O** [108] - Dynamics-aware method to learn from both online and offline data.

5. **DARC** [107] - Method based on reward adjustments with domain classifiers .

6. **DAP (Ours)** - Dual action policy predicting two set of actions (Sect. 3.4.1)

7. **DAP+U (Ours)** - DAP + **U**ncertainty-based Robust Action Resampling (Sect. 3.4.3)

Implementation details. We adopted the default hyperparameter settings from DARC [107] for our DAP implementation. All policies were trained for 1M steps. For the regularization term λ (Equation 3.6), we observed stability across values in the range $[0.05, 0.2]$, thus we fixed $\lambda = 0.10$ for all experiments, except for the ablation study of λ. The number of ensembles used to calculate uncertainty was set to $N = 5$ to achieve a good balance between accuracy and speed. The scaling parameter for uncertainty-based action resampling (Equation 3.7) was set to $k = 0.10$ for all experiments except for the ablation study of k.

3.5.2 Main Results

We evaluated baseline policies across four target tasks. Each evaluation comprised 100 episodes across 3 random seeds per checkpoint. The evaluation curves are presented in Figure 3.3, with the title specifying details about the tasks and environment settings. We plot target environment returns against the number of source environment training steps. For "RL on source" and "RL on target", we show their respective maximum returns as horizontal dashed lines. For all tasks, "RL on source" performs significantly worse than "RL on target". The huge gaps suggest that policies optimized for the source domain are not directly transferable to the target domain.

Our experiments reveal several key findings. First, H2O performs poorly in all tasks, often worse than "RL on source". This is because H2O relies on access to optimal target data for good performance. Next, without online interaction, both DARC and GARAT struggle, achieving performance slightly better or worse than "RL on source." Our proposed method, DAP, outperforms all other baselines significantly. It achieves at least double the return compared to "RL on source" on all tasks, demonstrating its effectiveness despite utilizing only sub-optimal offline data. Furthermore, DAP with uncertainty estimation (DAP+U) shows additional improvement on most tasks, nearly matching the performance of "RL on target" in some cases. This demonstrates the effectiveness of the robust action resampling. Notably, the half-cheetah task remains the most challenging, with even the best method achieving only ~ 3500, significantly lower than the optimal score of 9000.

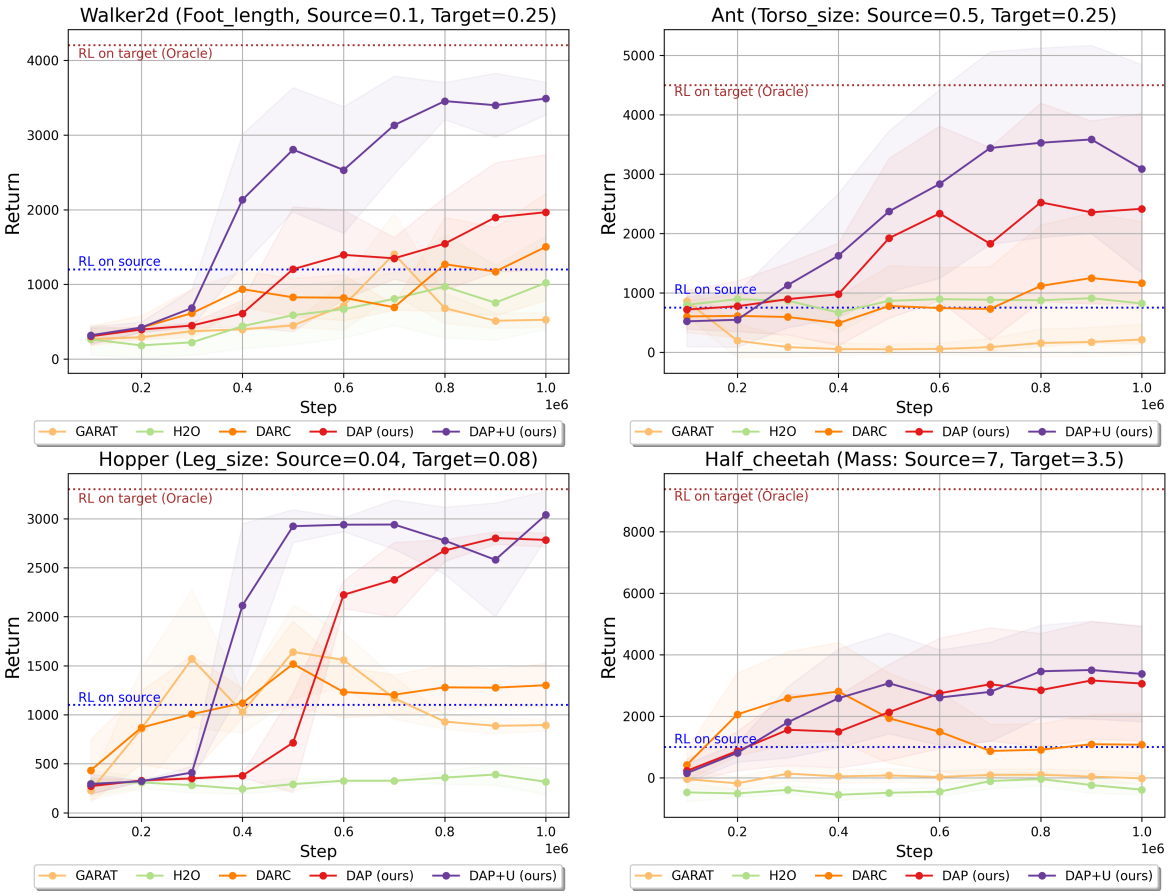


Figure 3.3: **Main results evaluated in target environment.** We compare our proposed methods, DAP and DAP+U, against several baseline approaches.

3.5.3 Ablation Experiments

Regularization Effects. The interplay between the source and target policies in DAP is crucial to its performance. The source policy serves a dual purpose: optimizing returns and generating state sequences indistinguishable from the target tasks. Meanwhile, the target policy primarily influences the reward shaping term. The regularization parameter λ plays a pivotal role in maintaining the balance between these two policies. Our ablation studies on λ as seen in Figure 3.4 revealed a clear trade-off: as λ approaches 0, performance on the target task significantly deteriorates due to the target policy generating "interesting" actions without optimizing returns. Conversely, large λ values cause DAP to converge towards DARC-like behavior, as the regularization term dominates the shaping. We found an optimal range for λ that preserves DAP's unique benefits while ensuring sufficient closeness between the policies. This balance

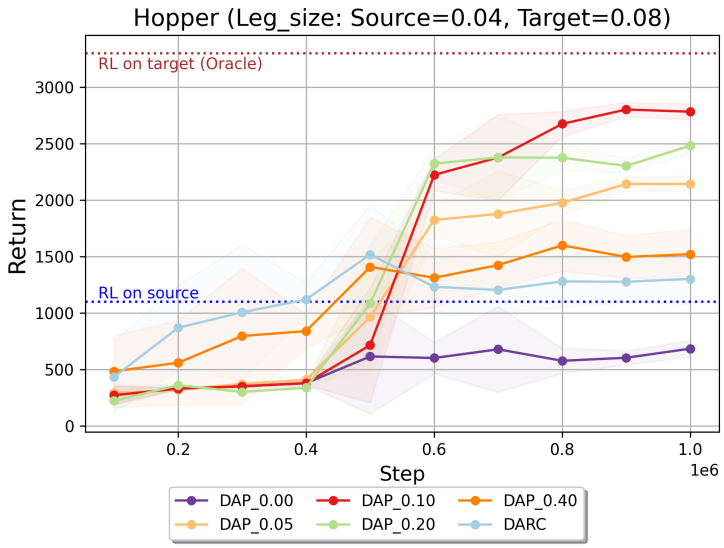


Figure 3.4: **Ablation: Regularization Effects on parameter λ.**

is critical: it allows the target policy to explore potentially beneficial actions while still leveraging the source policy's optimized behavior. The careful tuning of λ thus enables DAP to outperform both unconstrained exploration and strict imitation of the source policy.

Uncertainty-based action resampling. Our next experiment investigates the importance of the scaling parameter, k, for uncertainty-based action resampling (Equation 3.7). We evaluate a range of k values on the Walker2D environment and present the results in Figure 3.5. We begin with DAP without any resampling ($k = 0$) as the baseline. When we slightly increase k to 0.01, we observe slight improvement over the baseline at certain points during training. However, this performance gain doesn't persist throughout the training process. Further increasing k to 0.05 and 0.10 leads to significant improvement over the baseline, with $k = 0.10$ achieving the best performance. However, setting k to larger values like 0.15 and 0.20 results in a deterioration of the policy's performance. Notably, at $k = 0.20$, the policy exhibits instability and fails completely.

Size of target dataset. Our next experiment investigates the impact of the target dataset size, M. We evaluate a range of M values on the Walker2D environment and present the results in Figure 3.6. The policy setting is DAP+U (DAP with uncertainty estimation, $k = 0.10$). We

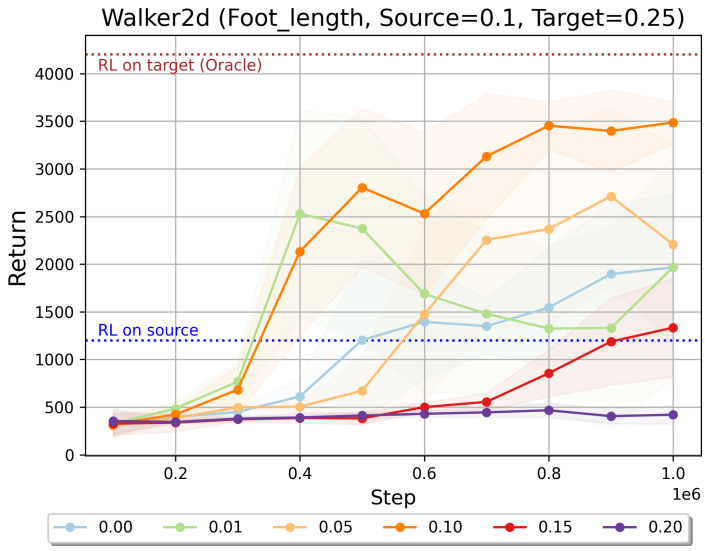


Figure 3.5: **Ablation: Effect of scaling parameter for uncertainty-based action resampling, k.**

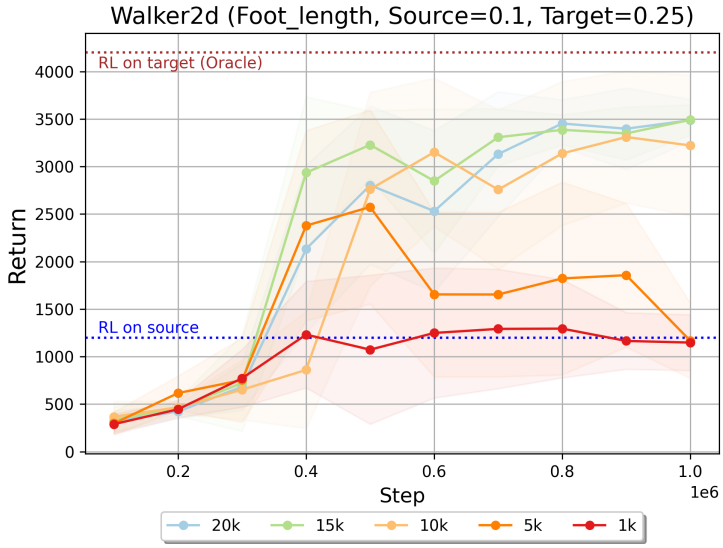


Figure 3.6: **Ablation: Effect of size of offline dataset, M, collected in target environment.**

begin with M = 20,000 as the baseline similarly used in the main experiment. We observe that decreasing M to 15,000 and 10,000 has minimal impact on the results, with the evaluation curves remaining relatively similar throughout the training process. However, a further decrease to M = 5,000 shows a decline in performance beyond 400,000 steps. Interestingly, with a bare minimum of M = 1,000 samples (equivalent to 1 episode), the policy maintains performance close to "RL on source" and avoids complete failure.

3.6 Conclusion

We propose the Dual Action Policy, a method for reinforcement learning that tackles the reality gap. Unlike conventional approaches, DAP employs a single policy to predict two sets of actions simultaneously. The first set maximizes the task reward, while the second set addresses the dynamics mismatch through reward adjustments. This decoupling makes it easier to maximise the overall reward in the source domain during training. To improve robustness, we introduce additional exploration based on uncertainty in dynamics estimation. Through action resampling, the agent explores areas with high dynamics uncertainty during training, enhancing its robustness during deployment. Experimental results demonstrate the effectiveness of DAP in achieving superior performance compared to several strong baselines across diverse and challenging target environments. Notably, DAP with uncertainty-based exploration (DAP+U) further improves performance, nearly matching the Oracle performance in some cases.

Chapter 4

Mitigating Dynamics Mismatch in Offline Reinforcement Learning

4.1 Introduction

Offline RL offers a valuable alternative to simulation-based training [37, 126], complementing the benefits of online RL explored in Chapter 3. This approach utilizes pre-existing datasets, eliminating the need for real-time interaction with the environment during training. By analyzing and extracting knowledge from these datasets, offline RL algorithms learn effective control policies. A significant advantage of offline RL is its ability to leverage diverse datasets from multiple sources, potentially including expert demonstrations or high-performing agents. This allows the RL agent to learn from a vast array of experiences that may be challenging to replicate in simulations. Consequently, offline RL not only addresses safety and feasibility concerns but also opens up new avenues for training intelligent agents in various fields. For instance, offline RL can be employed to train surgical robots by learning from demonstrations by experienced surgeons [47, 48], or to enhance data-driven financial trading by analyzing historical market data [49, 50]. In medicine, offline RL can improve diagnosis by learning from past patient records and clinical data [51, 52]. Additionally, it holds potential for revolutionizing autonomous vehicle development by learning from vast amounts of driving data collected from real-world scenarios [53–55].

However, offline RL is not without its challenges. The effectiveness of offline RL algorithms is heavily dependent on the quantity and quality of the available data. In scenarios where data is scarce or of low quality, the performance of offline RL can deteriorate significantly [127, 128]. Therefore, despite the advantage of not requiring live interaction with the

environment, a fundamental challenge in offline RL lies in the acquisition of large-scale, high-quality datasets, particularly those containing expert-level demonstrations. Addressing this challenge is crucial for realizing the full potential of offline RL in a wide range of real-world applications.

This work explores the potential of leveraging more accessible source datasets to address the challenge of data scarcity in offline RL. We draw inspiration from the success of transfer learning in supervised learning, where the utilization of data from multiple sources, including those that are easier to obtain, has become increasingly common [129–133]. We aim to adapt this principle to offline RL, where the scarcity of high-quality data can significantly limit the learning process. Specifically, we propose to incorporate readily available *off-dynamics* source datasets into the offline RL framework. These datasets are characterized by their alignment with the target task's objective, meaning they contain relevant information about the desired behavior or outcome. Although these datasets may be collected from similar robots, subtle dynamic variations can exist between individual units due to factors like manufacturing inconsistencies, wear and tear, or environmental conditions. This can introduce dynamics mismatch, potentially impacting the performance of RL agents when transferred to the target robot.

This strategy can be well applicable to real-world settings. For instance, in the domain of self-driving cars, data collected from various cities or different vehicle models, even if they possess distinct dynamics due to varying road conditions or vehicle specifications, can still be leveraged to improve the learning process. Similarly, in medical diagnosis, data from similar but not identical illnesses can provide valuable insights and contribute to the development of more accurate diagnostic models. In the field of surgical robotics, initial training on artificial organs can offer a valuable stepping stone for learning complex surgical procedures before transitioning to real patients. Furthermore, financial trading algorithms can benefit from utilizing data from larger, related markets to inform their decision-making processes in smaller markets with limited historical data.

Several approaches have been proposed to leverage off-dynamics samples to boost RL training. One popular method is to introduce a reward bonus to incentivize agents for taking actions which end up resembling the target environment [107, 108, 113, 115, 128]. However, this incurs extra cost and renders the learning of the primary task to be less effectively due to the embedded nature of the reward bonus. An alternative method is to apply transformations to

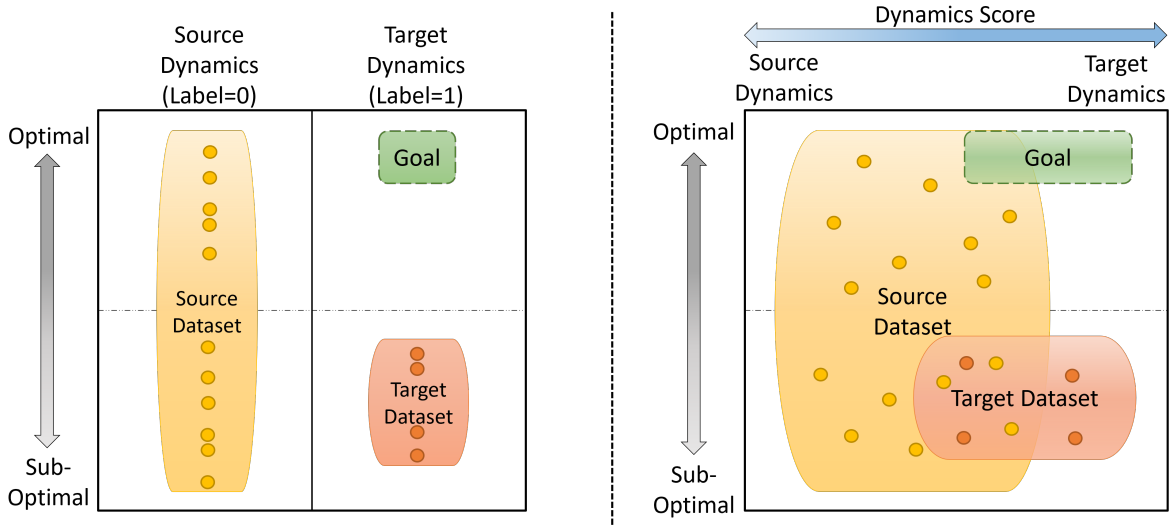


Figure 4.1: **Overview of our proposed framework for off-dynamics offline RL.** (Left) We utilize an accessible off-dynamics source dataset to enhance a limited target dataset for Offline RL. Our goal is to generate optimal trajectories within the green region. (Right) By conditioning a diffusion planner with our proposed continuous dynamics score, we enable the model to capture the underlying dynamics structure within the latent space through overlapping dynamics information.

the source domain [104–106], making it behave like the target domain. This allows the agent to learn as if it were in the target domain. However, this requires *online interactions* with the target environment, making it unsuitable for our problem setting.

To address these limitations, we propose to utilize the flexibility and expressiveness of *diffusion probabilistic models* (DPMs) [67–69] to learn a joint distribution of both source and target data for greater data efficiency. DPMs have demonstrated impressive capabilities in image generation [70, 72], audio generation [73], video generation [75] and more recently in Offline RL [84, 86, 87]. To enable our model to generate trajectories for the target environment, we utilize classifier-free guidance [78] by conditioning our model with dynamics-related contexts. For this context, we propose a *continuous dynamics score* as an alternative to the discrete dynamics labels as seen in Figure 4.1. This "soft score" allows for greater coverage, enabling overlap between trajectories from different datasets. Furthermore, we incorporate an inverse-dynamics context that measures the closeness to the target dynamics, ensuring the generated trajectories adhere to the target environment's dynamic constraints. Together, the context fa-

cilitates capturing the underlying dynamics structure within the latent space more effectively, leading to improved generation performance.

We conduct comprehensive experiments to evaluate the effectiveness of our method on a diverse set of challenging off-dynamics settings. Empirical results demonstrate that our method outperforms several strong baselines. The simplicity of the proposed dynamics-related contexts coupled with the powerful capabilities of DPMs allow us to effectively leverage an accessible off-dynamics source dataset. Lastly, we demonstrate that by modifying the context, we can interpolate between source and target dynamics, making the model more robust to subtle shifts.

4.2 Related Work and Background

4.2.1 Problem Formulation: Off-Dynamics Offline RL

In this chapter, we aim to enhance a limited target dataset offline using an accessible off-dynamics source dataset. We go beyond the standard offline RL framework of using a single fixed static offline dataset $\mathcal{D}_{\text{target}}$. Formally, similarly defined in [128], we incorporate an additional source dataset $\mathcal{D}_{\text{source}} = (s, a, r, s')$, collected by another unknown behavior policy μ_{source}, where $\mu_{\text{source}} \neq \mu_{\text{target}}$. A key distinction is that the source dataset is derived from an easily accessible source MDP $\mathcal{M}_{\text{source}}$, which exhibits different transition dynamics from the target MDP $\mathcal{M}_{\text{target}}$, i.e., $\exists(s, a, s') : P_{\text{source}}(s'|s, a) \neq P_{\text{target}}(s'|s, a)$. We assume that μ_{source} is nearly optimal, and $\mathcal{D}_{\text{source}}$ sampled under $\mathcal{M}_{\text{source}}$ is abundant but off-dynamics. Conversely, the target dataset is presumed to have limited and suboptimal trajectories sampled under $\mathcal{M}_{\text{target}}$. Our goal is to enable the transfer of knowledge between the offline datasets $\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$ to diminish the data dependency of $\mathcal{D}_{\text{target}}$ for learning an optimal policy for $\mathcal{M}_{\text{target}}$. For brevity, we now refer the transition dynamics of the source and target domain as source dynamics and target dynamics respectively.

4.2.2 Dynamics Modelling for Dynamics Mismatch

Numerous works have explored inverse dynamics modeling to address the dynamics mismatch in sim-to-real transfer. A key idea is to learn a transformation that makes the dynamics learned in the simulator resemble those of the target environment. In early work, [134] proposed using inverse dynamics modeling to align the dynamics learned in simulation with those of the target

environment. This is achieved by sampling actions in the source environment and then utilizing an inverse target dynamics model to determine corresponding actions. The inverse dynamics are modeled using a neural network, assuming some data collection in the target environment. Building on this concept, Grounded Action Transforms (GAT) [104] also models inverse target dynamics but relaxes the assumption of known forward source dynamics. Instead, GAT proposes collecting samples in the simulator to learn the forward source dynamics. Further improvements came with Reinforced GAT (RGAT) [105], where the inverse model is represented as a separate policy rather than a supervised model. This approach addresses the state distribution shift problem in the inverse modelling of GAT by directly predicting the output of the forward model. In Generative Adversarial Reinforced Action Transformation (GARAT) [125], a single discriminative model is learned to predict whether trajectories come from the source or target environment, inspired by Generative Adversarial Imitation Learning (GAIL) [135]. This eliminates the need for separate forward and inverse models. Overall, information extraction between source and target environments primarily involves inverse and forward dynamics models using regression or discriminative models using classification.

4.2.3 Off-dynamics Offline RL

In the online RL context described in Chapter 3, we introduced DARC, which uses two domain classifiers to measure the difference between source and target domains. This method encourages the agent to choose actions that more closely match the target dynamics. In offline settings, we can apply this concept to address dynamics differences, but we must also tackle the value overestimation issue in offline RL. Leveraging on this idea, Dynamics-Aware Reward Augmentation (DARA) [128] applies the DARC reward adjustment to all existing rewards in the offline dataset. This modification allows for subsequent policy training using any offline RL method that addresses value overestimation. A similar approach was taken by Niu et al. [108], who combined online and offline learning samples in a hybrid fashion. However, these methods using DARC reward adjustments face the similar limitations of the action set discussed in Section 3.4, where restricting same action sets for reward adjustments and policy updates during training can limit the search for an optimal solution that maximizes both rewards.

4.3 Approach

We present our novel approach to improve a limited, sub-optimal target dataset, $\mathcal{D}_{\text{target}}$, by leveraging a larger, diverse but off-dynamics source dataset, $\mathcal{D}_{\text{source}}$. We achieve this by training a conditional DPM to learn the joint distribution of both datasets. Naively training both datasets would be ineffective, as it would bias the model towards the source dynamics due to the larger size and optimality of $\mathcal{D}_{\text{source}}$. This bias would cause the generated trajectories not to align with the target environment's dynamics.

To address this challenge, the model is conditioned on the following two dynamics-related contexts:

- Continuous dynamics score: This replaces discrete labels (Figure 4.1) with a "soft" score, allowing for smoother transitions and overlap between trajectories from different datasets.
- Inverse-dynamics context: This measures how closely generated trajectories align with the target environment's dynamics, ensuring the generated samples adhere to its specific constraints.

These contexts enable the model to learn the joint distribution while adhering the inherent dynamics within the latent space.

4.3.1 Dynamics Score Context

We aim to introduce a context that can effectively capture both the differences and similarities between $\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$ in terms of dynamics information. This context is crucial for understanding how the system behavior changes across different environments or conditions. A straightforward method is assigning the context as a discrete one-hot label, for example, using '0' for samples from $\mathcal{D}_{\text{source}}$ and '1' for those from $\mathcal{D}_{\text{target}}$. This approach provides a clear distinction between the two datasets, making it easy to identify which environment a particular sample comes from. However, this discrete labeling does not provide sufficient information due to the potential overlaps in dynamics between the datasets. The binary nature of this approach does not allow for representing partial similarities or gradual changes in dynamics, which are often present in real-world systems. For instance, there could exist transitions in both $\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$ that are identical, indicating a shared dynamic characteristic across both domains.

These similarities are important to recognize, as they represent aspects of the system behavior that remain consistent across different environments.

To address this limitation, we propose replacing the discrete context with a continuous score. This continuous representation allows for a more fine-grained description of the dynamics, capturing subtle differences and similarities. This score should also be symmetric, ensuring equal representation for trajectories originating from $\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$. The symmetry property is important as it prevents bias towards either the source or target domain, treating both datasets equally in the analysis. To achieve this, for each trajectory τ in the form of Equation 2.37, we define the dynamics score over horizon, H as

$$c_{\text{dyn\_score}}(\tau) = \frac{1}{\kappa H} \sum_{t=0}^{H-1} \{\log[P_{\text{target}}(s_t, a_t, s_{t+1}) + \epsilon] - \log[P_{\text{source}}(s_t, a_t, s_{t+1}) + \epsilon]\}. \tag{4.1}$$

In this equation, P_{source} and P_{target} represent the probability of a given transition originating from the source and target datasets respectively. These probabilities quantify how likely a particular state-action-next state transition is to occur in each environment. The parameter ϵ is a small value added to prevent infinities when taking logarithms, ensuring numerical stability in the calculations. The scaling parameter κ is introduced to keep the score within the range $[-1, 1]$, making it easier to interpret and use in subsequent analyses. Smaller score values correspond to trajectories with dynamics that more closely resemble the source dynamics, while larger score values correspond to trajectories with dynamics that more closely resemble the target dynamics. This allows for a continuous spectrum of similarity between the source and target domains. When the score is zero, it means both dynamics are equally likely, representing a point of shared characteristics or transition points between the two environments. The details of modeling P_{target} will be elaborated in Section 4.3.3, where we will discuss the practical aspects of implementing this scoring system.

4.3.2 Inverse-dynamics Context

When the diffusion sampling process attempts to maximize the out-of-distribution return context, the resulting trajectory may not fully comply with the dynamics constraints of the underlying target MDP. To enforce these constraints more effectively, one potential approach is to

employ an inverse action based on inverse dynamics, as demonstrated in [136]. This involves using a separately trained inverse dynamics model to predict the action that best aligns with the underlying MDP given two consecutive states.

Building upon this concept, we integrate inverse dynamics information into our method. However, a key distinction is that instead of applying the inverse action as a post-processing step after trajectory generation, we directly incorporate it as a context for the conditional model. This modification is motivated by the observation that if the inverse action is computed after the trajectory has already been generated, the consecutive states within the trajectory might have already violated the dynamics constraints, rendering the inverse model incapable of accurately recovering the correct action. By incorporating the inverse dynamics constraints directly into the training process, we compel the trajectory to adhere to these constraints as closely as possible during the conditional sampling process, thereby preventing violations of the dynamics constraints from occurring in the first place.

Formally, for each trajectory τ adhering to the structure defined in Equation 2.37, the inverse context, c_{inverse}, is computed as follows:

$$c_{\text{inverse}}(\tau) = \frac{1}{H}\sum_{t=0}^{H-1} \log[1 + \|I_{\text{target}}(s_t, s_{t+1}) - a_t\|_2], \tag{4.2}$$

where I_{target} is the target inverse dynamics model, H is the planning horizon. The logarithm function helps to compress the range of the large-valued errors.

4.3.3 Practical Algorithm

We begin by outlining the process of learning P_{target}, P_{source} and I_{inverse}. To model P_{target}, we parameterize a binary classifier, $p_\phi = p(s_t, a_t, s_{t+1}; \phi)$, with a multi-layer perceptron (MLP). For P_{source}, we simply use $1 - p_\phi$. We fit p_ϕ by minimizing the standard binary cross-entropy loss:

$$\begin{aligned} L(\phi) = -\mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}_{source}\cup\mathcal{D}_{target}} \\ \left[y \log(p_\phi) + (1-y)\log(1-p_\phi)\right], \end{aligned} \tag{4.3}$$

where $y = 0, 1$ represents the source and target labels respectively.

Next, to model the inverse dynamics I_{target}, we parameterize it by ψ with another MLP, and minimize the standard mean-squared error between the predicted action and true action a_t:

$$L(\psi) = \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}_{target}}\left[(I_{target}(s_t, s_{t+1}; \psi) - a_t)^2\right]. \tag{4.4}$$

Algorithm 2 Training: Off-dynamics Conditional Diffusion Planners

Input: Target dataset $\mathcal{D}_{\text{target}}$, Source dataset $\mathcal{D}_{\text{source}}$
Input: Number of training updates N,
        Number of diffusion time steps T
Initialize dynamics score model q_ϕ and inverse model f_ψ
Fit q_ϕ using Loss in Eq.4.3 over $\mathcal{D}_{\text{target}} \cup \mathcal{D}_{\text{source}}$
Fit f_ψ using Loss in Eq.4.4 over $\mathcal{D}_{\text{target}}$
// Start Conditional Diffusion Training
Initialize Conditional U-Nets ϵ_θ
for n in $1, 2, ..., N$ **do**
    Sample stratified batch $\tau_{\mathcal{B}} \in \mathcal{D}_{\text{target}} \cup \mathcal{D}_{\text{source}}$
    Compute context $c_{dyn\_score}(\tau_{\mathcal{B}}) = q_\phi(\tau_{\mathcal{B}})$
    Compute context $c_{inverse}(\tau_{\mathcal{B}}) = f_\psi(\tau_{\mathcal{B}})$
    Set full context $\boldsymbol{y}(\tau_{\mathcal{B}}) = [R(\tau_{\mathcal{B}}), c_{dyn\_score}(\tau_{\mathcal{B}}), c_{\text{inverse}}(\tau_{\mathcal{B}})]$.
    for t in $1, 2, ..., T$ **do**
        Update θ with $\epsilon_\theta(\tau_{\mathcal{B}}, t, \boldsymbol{y}(\tau_{\mathcal{B}}))$ using Loss in Eq.4.6
    end for
end for

Now, we introduce a practical algorithm that combines all components for training a off-dynamics conditional diffusion-based planner. Our model follows the classifier-free approach with input trajectories τ, following Equation 2.37. The models are conditioned on the full context $\boldsymbol{y}(\tau)$, which consists of the dynamics score, the inverse dynamics and the normalised return $R(\tau) \in [0, 1]$ as follows:

$$\boldsymbol{y}(\tau) = [R(\tau), c_{dyn\_score}(\tau), c_{\text{inverse}}(\tau)]. \tag{4.5}$$

This context summarizes the optimality and the dynamics information of each trajectory in a continuous form as movitated in Fig. 4.1. By leveraging dynamics information, the conditional model gains the ability to flexibly learn from both $\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$ with greater data efficiency. It also enables the conditional model to generate optimal trajectories from either domain in a seamless manner. The objective for the conditional diffusion process is,

$$\max_\theta \mathbb{E}_{\tau \sim (\mathcal{D}_{\text{source}} \cup \mathcal{D}_{\text{target}})} \left[\log p_\theta \left(\boldsymbol{x}_0(\tau) \mid \boldsymbol{y}(\tau) \right) \right]$$

with the loss given by

$$\mathcal{L}(\theta) = \mathbb{E}_{k \sim [1,K], x_0 \sim q, \epsilon \sim \mathcal{N}(0,I)} \left\| \epsilon - \epsilon_\theta(x_k(\tau), k, y(\tau)) \right\|^2 \tag{4.6}$$

The complete training procedure is detailed in Algorithm 2. During planning, we set the target context $y(\tau) = [1, 1, 0]$ to generate trajectories that maximize the reward for the target environment. This aligns with Equations 4.1 and 4.2, where $P_{\text{target}}(s_t, a_t, s_{t+1}) = 1$ and $I_{\text{target}}(s_t, s_{t+1}) = a_t$, ensuring the generated trajectories adhere to the target dynamics.

4.4 Experiments

We design and conduct comprehensive experiments to thoroughly compare the effectiveness of our method with existing ones. We begin by outlining the experimental setup, which involves creating a diverse and challenging set of off-dynamics datasets for training and evaluation. Nine distinct settings are used to compare our approach with several strong baselines. Additionally, an ablation study is conducted to investigate the specific contribution of each dynamics context to the performance improvement. Finally, we investigate the robustness and capacity of our model to seamlessly interpolate between source and target dynamics.

4.4.1 Experimental Setup

Datasets. We formulate our experimental datasets based on our proposed framework, illustrated in Figure 4.1 (left). We require a large and diverse offline source dataset, and select the Hopper, Walker2d, and Halfcheetah medium-expert datasets from D4RL [1], each consisting of around 2 million samples. For the target dataset, we need limited and sub-optimal samples with different dynamics compared to the source. To achieve this, we first create multiple off-dynamics variants by modifying parameters like mass, size, control range, friction, and gear torques within each environment. We proposed nine different experimental settings across three distinct environments, each with three variations in their physical properties as seen in Table 4.1. Subsequently, we collect 10,000 samples from each modified environment using a behavioral policy trained with Diffuser in the respective source environment. This diverse and challenging off-dynamics setting allows for comprehensive evaluation of our method.

Baselines. We benchmark against various data-driven control algorithms for off-dynamics offline RL. We apply the off-dynamics reward compensation method DARA [128] to several well-performing offline RL algorithms. Among these offline RL algorithms, we include

Task	Property	Source	Target	Environment
Half-cheetah	total mass	14	7	Half-cheetah_mass
	torso size	0.046	0.092	Half-cheetah_torso_size
	control range	[-1,1]	[-0.5,0.5]	Half-cheetah_ctrl_range
Walker2d	thigh action	Enabled	Disabled	Walker2d_thigh
	foot gear torque	100	70	Walker2d_foot_torque
	foot length	0.1	0.25	Walker2d_foot_len
Hopper	foot friction	2.0	1.7	Hopper_foot_fric
	torso stiffness	0	3	Hopper_torso_stiff
	leg size	0.03	0.04	Hopper_leg_size

Table 4.1: **Nine different experimental settings across three distinct environments.** Each with three variations in their physical properties. The columns 'Source' and 'Target' represent the dynamics settings of $\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$, respectively.

model-free methods like BCQ [57] and CQL [59], as well as diffusion-based methods like Diffuser [86]. In addition to DARA, we include the fine-tuning method, which performs 10,000 additional updates on a pre-trained source model using the target dataset.

Implementation details. For the hyper-parameters of DPM, we follow the default settings in Diffuser [86]. For both the classifier and inverse models, we use basic MLPs with 2 hidden layers and 32 nodes, each trained for 200k updates. We apply *min-max normalization* to the outputs of these models prior to context computations. During training, we apply a context *dropout* of $p = 0.5$ for each context independently. During sampling, we use a conditional sampling weight of $w = 0.9$ for all environments. We evaluate our models over 300 episodes across 3 seeds (total 900 episodes) and compute the normalized score over the target environments [1].

4.4.2 Main Results

The normalized scores obtained by various methods under three training scenarios are presented in Table 4.2: 1) training exclusively on source data, 2) pretraining on source data followed by finetuning on target data, and 3) joint training on both source and target data.

In the source-only training scenario, where a Diffuser model trained on optimal samples from the source dataset is evaluated on the target environment ("$\mathcal{D}_{\text{source}}$" column), we observe

a substantial decrease in performance across all environments. This finding underscores the significant challenges posed by the off-dynamics nature of our experimental environments, where the dynamics of the source and target environments differ. The scores achieved by the Diffuser in this setting serve as a baseline for evaluating performance on the target dataset, as the Diffuser policy is subsequently used to collect target data.

When we transition to the finetuning scenario ("$\mathcal{D}_{\text{source}}$ pretrain+$\mathcal{D}_{\text{target}}$ finetune" column), where models are pretrained on the source dataset and then finetuned on the target dataset, we find that all three methods (Diffuser, CQL, BCQ) generally underperform compared to the source-only Diffuser. This suggests that the finetuning process may struggle to strike an effective balance between learning the dynamics of the target environment while retaining the knowledge of optimal behaviors acquired from the source dataset.

Finally, in the joint training setting ("$\mathcal{D}_{\text{source}} \cup \mathcal{D}_{\text{target}}$ Joint Training" column), where models are trained on both source and target data simultaneously, we observe that both DARA methods face difficulties in this low-data regime, despite the dynamics-aware reward adjustment mechanism. In contrast, our proposed conditional DPM with dynamics contexts consistently outperforms all other baseline methods. This superior performance can be attributed to the powerful capability of the DPM to generate high-reward trajectories specifically tailored for the target environment, guided by the contextually relevant information about the dynamics of that environment. In the subsequent sections, we will analyze the individual contributions of each dynamic context to the overall performance improvement.

4.4.3 Contexts Analysis

To gain a more comprehensive understanding of each context's contribution to the performance of the conditional DPM model, we conducted additional experiments. The mean normalized score for each environment is presented in Figure 4.2. As a starting point, we consider the conditional DPM with a single context comprising solely the trajectory return (R, blue bar). This serves as a baseline to assess the impact of adding dynamics-related contexts.

When we augment the context with the dynamics score (R+DS, orange bar), we observe a notable improvement in performance across all environments, achieving a score of 63.8. Remarkably, the inclusion of just this single dynamics-related context is sufficient to surpass the

Environment	$\mathcal{D}_{\text{source}}$	$\mathcal{D}_{\text{source}}$ pretrain + $\mathcal{D}_{\text{target}}$ finetune			$\mathcal{D}_{\text{source}} \cup \mathcal{D}_{\text{target}}$ Joint Training		
	Diffuser	Diffuser	CQL	BCQ	CQL+DARA	BCQ+DARA	Proposed
Half-cheetah_mass	30.2	32.6	26.3	28.7	27.3	19.8	**56.8** ± 5.9
Half-cheetah_torso_size	43.6	38.2	32.0	46.8	45.9	32.2	**54.1** ± 7.9
Half-cheetah_ctrl_range	41.7	25.2	52.2	2.2	53.2	53.1	**61.5** ± 6.6
Walker2d_thigh	22.6	39.1	28.8	-0.3	32.2	10.2	**58.5** ± 9.0
Walker2d_foot_torque	42.9	36.7	46.8	51.0	-0.2	44.2	**63.4** ± 12.2
Walker2d_foot_length	42.7	41.6	48.3	43.6	-0.2	5.3	**63.0** ± 10.7
Hopper_foot_fric	50.7	52.5	32.1	40.2	49.7	109.5	**124.2** ± 31.5
Hopper_torso_stiffness	73.9	72.2	65.7	72.5	51.8	42.9	**95.0** ± 6.9
Hopper_leg_size	74.2	57.3	36.8	82.8	75.6	93.1	**95.9** ± 9.3
Average	46.9	43.9	41.0	40.8	37.3	45.6	**74.7**

Table 4.2: **Mean normalized scores evaluated on the target environment over 900 episodes across 9 diverse settings.** Our proposed method is a conditional diffusion planner with contexts according to Equation 4.5, trained with Algorithm 2.

performance of all other baseline methods listed in Table 4.2. This highlights the substantial impact that even a simple dynamics-related signal can have on guiding the DPM towards generating trajectories that are more aligned with the target environment dynamics.

Further incorporating the inverse dynamics context (R+DS+ID, green bar) yields additional performance gains, with a particularly significant improvement observed in the Hopper environment. This result suggests that even when the dynamics score effectively biases the generated trajectories towards the target environment, the inverse dynamics context provides supplementary guidance for the sampling process, resulting in trajectories that adhere more closely to the target dynamics.

To further investigate the role of inverse dynamics, we conducted an ablation study (R+DS+IA, red bar) in which we applied the inverse action (as discussed in Section 4.3.2) directly to the R+DS context. However, this modification led to a decline in performance compared to the model without the inverse action. This finding supports our hypothesis that the consecutive states generated by R+DS may already violate the dynamics constraints of the target environment, rendering the inverse model incapable of reliably recovering the correct action. These results collectively underscore the importance of carefully selecting and combining context information to optimize the performance of the conditional DPM model. The dynamics score plays a crucial role in guiding the model towards the target environment dynamics, while the inverse dynamics context can provide further refinements.

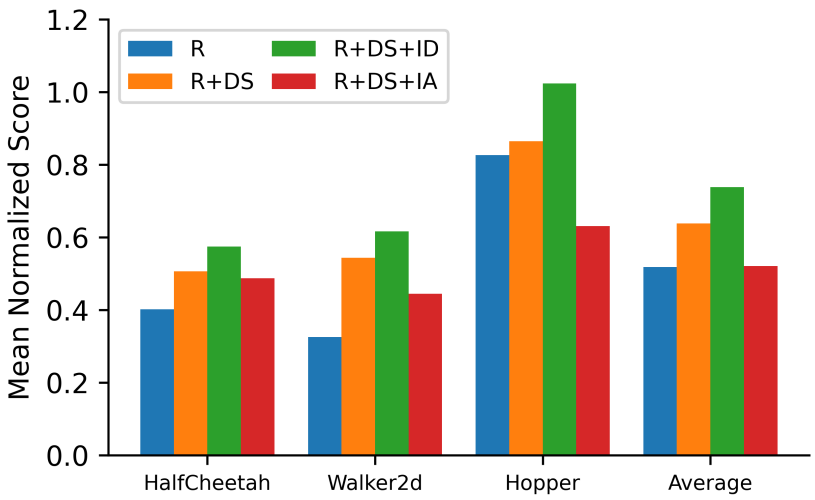


Figure 4.2: **Ablation: models trained with different contexts following Algorithm 2.** We report the mean normalized score across different settings per environment. ('R', blue) represents the base model conditioned on only the return. ('R+DS', orange) adds on the dynamics score as contexts. ('R+DS+ID', green) further adds on the inverse-dynamics context. ('R+DS+IA', red) applies inverse action on 'R+DS'.

4.4.4 Robustness

Inspired by the success of diffusion models in other domains, where they achieve smooth interpolation within the latent space by controlling contexts, we investigate whether our model exhibits similar behaviors. This ability becomes crucial in real-world scenarios, as target dynamics often undergo subtle shifts. The capacity to generalize across a range of dynamics without explicit training data for each variation is a key advantage for robust and adaptable reinforcement learning systems.

To explore this, we leverage one of our experimental settings involving Halfcheetah with varying masses (Table 4.2, first row). The source and target datasets correspond to masses of $m = 14$ and $m = 7$, respectively. This setup allows us to examine how well our model can interpolate between known dynamics and extrapolate to unseen conditions. Using the models trained with these two datasets, we further evaluate the masses in between (interpolation: $8 \leq m \leq 13$) and beyond (extrapolation: $3 \leq m \leq 6$). This comprehensive evaluation across a spectrum of masses enables us to assess the model's generalization capabilities and its ability to handle dynamics shifts of varying magnitudes.

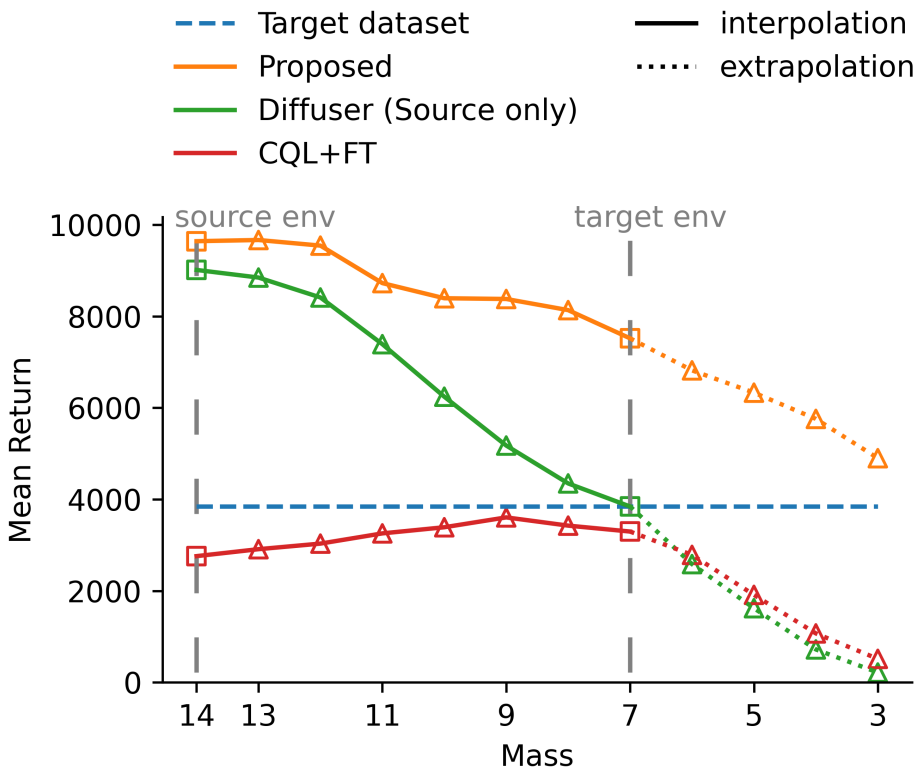


Figure 4.3: **Plot of the generalisation capabilities for Halfcheetah.** Models are trained on $\mathcal{D}_{\text{source}}$ ($m = 14$), $\mathcal{D}_{\text{target}}$ ($m = 7$). Models are evaluated at interpolated masses $8 \leq m \leq 13$ and extrapolated masses $3 \leq m \leq 6$. Mean returns are shown due to varying normalizing score factors across different masses.

We evaluate our model, alongside two baselines: (1) Diffuser, using only source data; (2) CQL+FT, source pre-train with target fine-tuning. These baselines represent alternative approaches to handling varying dynamics, providing context for assessing our model's performance. For our approach, we apply a simple linear scale on the dynamics-related contexts based on mass for intermediate evaluations. During extrapolation $3 \leq m \leq 6$, we utilize the context corresponding to the target mass $m = 7$. This strategy allows us to test how well our model can leverage learned representations to handle unseen dynamics. Figure 4.3 presents the results of our comparative analysis. Our method (orange line) exhibits robust performance across the interpolated range ($8 \leq m \leq 13$), even without training data following these environment's dynamics. This demonstrates the model's ability to generalize effectively within the range of known dynamics. Compared to CQL+FT (red line), our model maintains similar

performance for $m = 8, 9$, but exhibits a gradual decline beyond those values. This aligns with the expectation that the fine-tuned model has forgotten most of its source information. The comparison highlights the trade-off between adaptation to new dynamics and retention of previously learned knowledge. While sharing a similar architecture, the Diffuser trained on source data (green line) experiences a rapid performance drop beyond the source mass of $m = 7$. This suggests that the dataset with matching dynamics, despite its small quantity, remains crucial for effectively training our model. The contrast in performance underscores the importance of incorporating target domain data, even in limited quantities.

Finally, when extrapolating masses outside the range of the training data ($3 \leq m \leq 6$), all three models exhibit similar limitations in their capabilities. This is evident in the similar rate of performance degradation observed for all models in this region. The results indicate that while our model shows improved interpolation abilities, extrapolation to significantly different dynamics remains challenging across all approaches. These findings provide valuable insights into the generalization capabilities of our model and its potential for handling dynamic shifts in real-world applications. They also highlight areas for future research, such as improving extrapolation performance and developing more sophisticated context adaptation techniques. We have also provided qualitative results in https://youtu.be/7x7XVROjhR0, showcasing the performance of policies trained on datasets with 1× and 2× torso scales when applied to previously unseen Half-Cheetah torso sizes (1.3× and 1.7×).

4.5 Conclusion

We propose a new approach, which utilizes a conditional DPM with dynamics-related contexts to address the challenge of data scarcity in offline RL. We introduce a continuous dynamics score and an inverse-dynamics context to effectively capture the underlying dynamics structure within the latent space, enabling the model to learn from both a larger off-dynamics source dataset and a limited, sub-optimal target dataset. Experimental results demonstrate that our method significantly outperforms various baselines. Ablation studies further reveal the critical role of each dynamics context in improving performance. Additionally, our model exhibits promising robustness in handling interpolation scenarios, showcasing its potential for real-world applications with dynamic shifts in the target environment. As future work, we aim to explore the use of multiple off-dynamics datasets and validation in real-world applications.

Part II

From Personalization to Crowd Navigation: A Human-Centered Approach

Chapter 5

Personalization of Decision-making Systems

5.1 Introduction

In today's increasingly automated world, personalization is crucial for decision-making systems to effectively cater to individual's needs, preferences, and circumstances. Tailoring experiences enhances the system effectiveness and user satisfaction across diverse applications, such as customizing self-driving vehicles [137, 138], transforming robotic assistants into adaptive companions [139–143], and optimizing prosthetics for wearers' unique requirements [144–146]. However, accurately capturing and aligning the abstract and dynamic human preferences with automated systems remains a complex challenge [147–149].

This work tackles this personalization challenge in automated decision-making systems, aiming to create adaptable and reusable models that cater to individual user's needs [150–153]. While large-scale pretrained models offer broad capabilities [154–157], they lack the individual customization, and training personalized models for every user is infeasible. In contrast, it is more promising to first **pretrain** a model on large-scale **offline** data, and then **align** it with human preferences using smaller, user-specific preference datasets, as shown in Figure 5.1. The adoption of pretrained models from offline data avoids costly or risky real-time interaction, enabling broader applications in challenging environments [47, 48, 54]. The process for adapting human preferences must be **resource-efficient**, enabling both real-time updates for millions of users and rapid deployment on edge devices, while minimizing data requirements due to privacy concerns. Therefore, we adopt this pretrain-align framework to achieve personalized decision making.

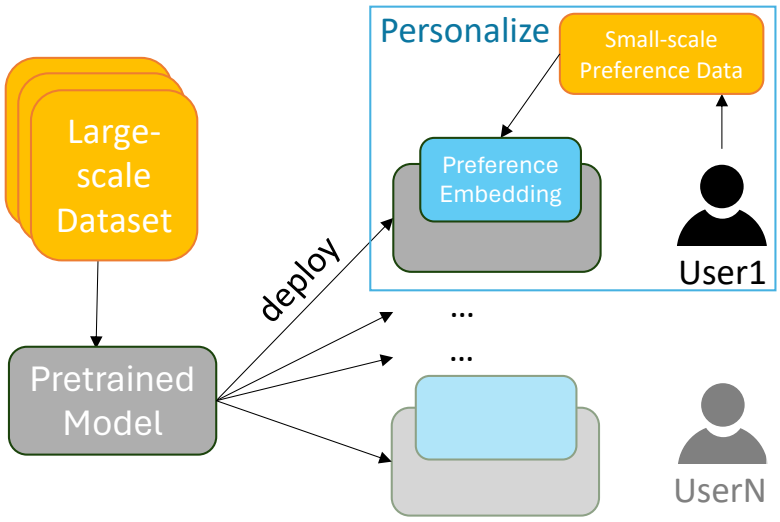


Figure 5.1: **Overview of personalizing decision-making models.**

However, there are still a couple of difficulties to realize this system. (1) For pretraining the decision-making models, some approaches [158, 159] perform the training without rewards, but require the online interaction with the environment, which is not applicable in our offline setting. Other approaches with offline RL [37, 59, 160, 161] relies on rewards, which are often unavailable or difficult to quantify for human preferences. It is important but challenging to address both requirements simultaneously. (2) For adapting models to human preferences, *Reinforcement Learning from Human Feedback* (RLHF) [148] has emerged as a key technique for integrating human preferences into decision-making systems [162–165]. It works by first learning reward models to capture individual preferences and then refining policies based on those learned reward models. *Direct policy optimization* (DPO) offers an alternative approach by directly aligning policies with human preferences, bypassing the need for a separate reward model [166]. However, both RLHF and DPO face computational challenges due to the large number of parameters involved during alignment, making them less resource-efficient. Furthermore, both methods require careful tuning to prevent the adapted model from deviating too far from the base model.

We propose a new pretrain-align framework to enable efficient and rapid personalized decision-making. Our solution is built atop of diffusion-based planners [84, 86], which leverage the expressive power of diffusion models [67, 69] to learn flexible and tractable models for

trajectory generation. Specifically, in the **pretraining phase**, we learn a conditional diffusion model from a large offline dataset consisting of reward-free trajectories.

We introduce *preference latent embeddings* (PLE), low-dimensional vectors that effectively encode human preferences, for rapidly adapting pretrained models to individual user preferences. During the **alignment phase**, when deploying the model for individual users, we adapt the pretrained conditional diffusion model without modifying its underlying parameters. This is achieved by using a novel inversion method that optimizes a learnable PLE on a small preference dataset. The compact representation of PLE enables seamless integration into decentralized systems and enhances real-world applicability. Results demonstrate that our method adapts more accurately to human preferences with less data, in both simulated and real-world scenarios.

5.2 Related Work

Inversion for Image Manipulation. In the domain of *generative adversarial networks* (GANs) [167], manipulating images often involves finding the corresponding latent representation of a given image, a process known as *inversion* [168,169]. This can be achieved through optimization-based techniques [170–172], which directly optimize a latent vector to recreate the target image when passed through the GAN, or through the use of encoders [173–175]. Similarly, in the domain of DPMs, inversion enables image manipulations such as cross-image interpolations and semantic editing in *DALL-E 2* [176]. Additionally, the concept of *textual inversion*, introduced in [177], allows for the representation of visual concepts using novel tokens within the embedding space of a frozen text-to-image model, resulting in personalized token embeddings.

Preference Learning. It has proven to be effective to leverage relative human judgments through pairwise preference labels for optimizing human preferences without direct access to the reward function. This approach shows significant success in various natural language processing tasks, such as translation [178], summarization [179, 180], story-telling [180], and instruction-following [32, 181]. It typically learns a reward function using a preference model like the Bradley-Terry model [182], and subsequently trains the model using RL algorithms [41, 43] to maximize the learned reward. *Direct policy optimization* (DPO) has been proposed as an alternative to directly align the policy with human preferences and learn from collected

data without a separate reward model [166]. DPO variants [183–186] have shown great alignment with human preferences that matches or surpasses reward-based methods. In the domain of RL, learning policies from preferences has been studied, as designing a suitable reward function can be challenging. Various approaches have been proposed [148, 158, 162–165] that learn a reward function from trajectory segment pairs. Recently, DPO has also been incorporated into this domain [187].

5.3 Background

Preference Modelling. Optimizing for human preferences in the generation of data samples x under specific conditions c, without explicit knowledge of the underlying reward function $r(c,x)$, poses a significant challenge in machine learning. A widely adopted approach to address this challenge involves defining a pairwise preference relationship between samples annotated by humans, where the winning sample x^w is preferred over losing sample x^l.

The Bradley-Terry model offers a framework for modeling and understanding these human preferences. It proposes that the probability of one sample being preferred over another can be expressed as a function of the difference in their underlying reward values. This can be formulated as:

$$p_{\mathrm{BT}}(x^w \succ x^l|c) = \sigma(r(c,x^w) - r(c,x^l))$$

where σ represents the sigmoid function, which maps the difference in reward values to a probability between 0 and 1.

In practice, we can leverage this model to estimate the hidden reward function by parameterizing it with a neural network denoted as ϕ. We can then train this network using maximum likelihood estimation on a fixed dataset consisting of prompts x and corresponding data pairs y_u, y_l, where the labels have been annotated by humans. The loss function for this binary classification task can be expressed as:

$$L_{\mathrm{BT}}(\phi) = -\mathbb{E}_{x,y_u,y_l}\left[\log\left(\sigma(r_\phi(x,y_u)) - \sigma(r_\phi(x,y_l))\right)\right] \tag{5.1}$$

By minimizing this loss function, we can learn the parameters of the neural network ϕ that best approximate the hidden reward function. This learned reward function can then be utilized

as feedback to guide the generation of data samples that align with human preferences, even without direct access to the true underlying reward function.

Reinforcement Learning from Human Feedback Reinforcement Learning from Human Feedback (RLHF) [32, 179] aims to refine a conditional distribution, $p_\theta(y_0|x)$, where x is drawn from a dataset D_x, to maximize the underlying reward function $r(x, y_0)$. This process also aims to regulate the Kullback-Leibler (KL) divergence concerning a base distribution p_{re}.

In essence, RLHF seeks to find the distribution $p_\theta(y_0|x)$ that generates outputs y_0 given inputs x that are not only highly rewarded by the latent reward function but also do not deviate significantly from a reference distribution p_{re}. This reference distribution can be thought of as a prior belief about the desired output distribution, and the KL divergence term acts as a regularizer, preventing the learned distribution from straying too far from this prior.

Formally, this optimization problem can be expressed as follows:

$$\max_{p_\theta} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_x, \boldsymbol{y}_0\sim p_\theta(\boldsymbol{y}_0|\boldsymbol{x})} \left[r(\boldsymbol{x}, \boldsymbol{y}_0)\right] - \beta\mathbb{D}_{\mathrm{KL}} \left[p_\theta(\boldsymbol{y}_0|\boldsymbol{x})||p_{\mathrm{ref}}(\boldsymbol{y}_0|\boldsymbol{x})\right] \tag{5.2}$$

In this equation, the first term represents the expected reward under the learned distribution p_θ, while the second term measures the KL divergence between p_θ and the reference distribution p_{re}. The hyperparameter β controls the strength of this regularization term, determining the trade-off between maximizing the reward and staying close to the prior distribution. A larger β value places a stronger emphasis on staying close to the reference distribution, while a smaller value allows for greater flexibility in maximizing the reward, potentially at the cost of increased divergence from the prior.

5.3.1 Direct Preference Optimization

Direct Preference Optimization (DPO) [166] is an approach that aims to optimize the generation of data samples to align with human preferences without explicitly learning a reward function. This method starts with a reference generative model, which may not initially incorporate human preferences, and has a conditional probability distribution denoted as $p_{\mathrm{ref}}(x|c)$. DPO then proceeds to fine-tune this conditional probability distribution into $p_\theta(x|c)$ by leveraging human feedback in the form of pairwise comparisons.

The core idea behind DPO is to adjust the model's parameters θ to increase the probability of generating samples that are preferred by humans. This is achieved by minimizing a loss

function that encourages the model to assign higher probabilities to preferred samples (x^w) and lower probabilities to less preferred samples (x^l), given a specific condition c. The loss function used in DPO is as follows:

$$L_{\text{DPO}}(\theta) = -\mathop{\mathbb{E}}_{c,x^w,x^l}\left[\log\sigma\left(\beta\log\frac{p_\theta(x^w|c)}{p_{\text{ref}}(x^w|c)} - \beta\log\frac{p_\theta(x^l|c)}{p_{\text{ref}}(x^l|c)}\right)\right]$$

In this equation, β serves as a hyperparameter that controls the extent to which the model is allowed to deviate from the reference policy. A larger value of β encourages the model to stay closer to the reference policy, while a smaller value allows for more flexibility in incorporating human preferences.

Building upon the principles of DPO, Identity Preference Optimization (IPO) [183] proposes a stronger constraint to ensure even closer alignment with the reference policy. IPO achieves this by minimizing a modified loss function:

$$L_{\text{IPO}}(\theta) = \mathop{\mathbb{E}}_{c,x^w,x^l}\left[\left(\log\frac{p_\theta(x^w|c)p_{\text{ref}}(x^l|c)}{p_\theta(x^l|c)p_{\text{ref}}(x^w|c)} - \frac{1}{2\beta}\right)^2\right] \tag{5.3}$$

This modified loss function effectively penalizes larger deviations from the reference policy, ensuring that the fine-tuned model adheres more closely to the initial preferences encoded in the reference model while still incorporating human feedback to improve its performance.

5.3.2 DPO for Diffusion Planners

Previous approaches derive an objective for general diffusion models with Direct Preference Optimization (DPO) for image generation [188]. This method combines the strengths of diffusion models with preference learning, enabling the generation of images that better align with human preferences. By adapting this approach to diffusion planning, we can formulate a loss function that incorporates preference information into the trajectory generation process. The loss function for Diffusion-DPO in the context of planning is derived as:

$$\begin{aligned} L_{\text{Diff-DPO}}(\theta) = \mathop{\mathbb{E}}_{\epsilon^w,\epsilon^l,\tau^w,\tau^l,k} \log\sigma\big(-\beta\big(&\|\epsilon^w - \epsilon_\theta(\tau_k^w,k)\|^2 - \|\epsilon^w - \epsilon_{\text{ref}}(\tau_k^w,k)\|^2 \\ &-(\|\epsilon^l - \epsilon_\theta(\tau_k^l,k)\|^2 - \|\epsilon^l - \epsilon_{\text{ref}}(\tau_k^l,k)\|^2)\big)\big). \end{aligned} \tag{5.4}$$

In this formulation, the terms τ^w and τ^l represent trajectories classified as 'winning' and 'losing' respectively, based on human preferences. The variables ϵ^w and ϵ^l represent the noise

associated with these trajectories at the k-th diffusion timestep (denoted by k). The term ϵ_θ refers to the fine-tuned diffusion-based planning model, while ϵ_{ref} denotes the frozen pre-trained model. The hyperparameter β is a scaling factor used to control the strength of the preference signal. The winning and losing trajectories, τ^w and τ^l, are sampled from a preference dataset D, where each trajectory is annotated by a human evaluator indicating whether it is considered 'winning' (preferred) or 'losing' (less preferred).

Following a similar approach to that employed for DPO, we can derive a loss function adapted for IPO within the context of diffusion planning. This loss function is formulated as:

$$L_{\text{Diff-IPO}}(\theta) = \mathop{\mathbb{E}}_{\epsilon^w,\epsilon^l,\tau^w,\tau^l,k}\Big[\big((\|\epsilon^l - \epsilon_\theta(\tau^l_k,k)\|^2 - \|\epsilon^l - \epsilon_{\text{ref}}(\tau^l_k,k)\|^2) - (\|\epsilon^w - \epsilon_\theta(\tau^w_k,k)\|^2 - \|\epsilon^w - \epsilon_{\text{ref}}(\tau^w_k,k)\|^2) - \frac{1}{2\beta}\big)^2\Big] \tag{5.5}$$

In both loss functions, Equation 5.4 (for Diff-DPO) and Equation 5.5 (for Diff-IPO), we observe four L2 error terms. Each of these terms corresponds to the loss of the DPMs as defined in Equation 2.29. Both functions aim to reduce the error of the fine-tuned model ϵ_θ on preferred trajectories τ_w while increasing the error on less preferred trajectories τ_l. This mechanism effectively increases the probability of generating preferred trajectories $p_\theta(\tau_w)$ and decreases the probability of generating less preferred ones $p_\theta(\tau_l)$, aligning the model's output with human preferences.

5.4 Methodology

To enable rapid adaptation of our pretrained model to individual user preferences, we introduce the concept of ***preference latent embeddings*** (PLE), denoted by z. PLEs are low-dimensional vectors that encode human preferences efficiently. Our method comprises three distinct stages, as illustrated in Figure 5.2: **pretraining, adaptation, and generation**. During the pretraining stage, we train the diffusion model without reward supervision and initialize a placeholder for the PLE. This step establishes a general-purpose generative model with a broad understanding of the task domain, prior to any specific preference adaptation. The adaptation stage follows, where we utilize a small set of human preference labels to partially fine-tune the pretrained model. This process allows us to identify the PLE that aligns with the given preferences, effectively capturing the user's specific requirements in our low-dimensional representation.

Once adaptation is complete, we enter the generation stage. Here, we leverage the identified PLE to generate trajectories that match the human preferences encoded within it. We now elaborate on each of these stages in detail.

5.4.1 Pretraining with Masked Trajectories

The pretraining stage of our method addresses two concurrent goals: training a general-purpose generative model that comprehensively understands the task domain without reward supervision, and initializing a meaningful representation for the PLE placeholder. To achieve the first goal, we employ the decision diffuser [84], which excels in training on offline trajectory datasets without explicit reward requirements. Its ability to incorporate additional context is crucial for our second goal.

For the second goal, we posit that each preference correlates with a set of similar trajectories. Consequently, we aim to map similar trajectories to similar embeddings to give structure for the PLE placeholder. We propose a learnable mapping $f : \mathbb{R}^{L\times(S+A)} \to \mathbb{R}^{d_e}, f(\tau_{\text{full}}) = z$, where τ_{full} is the corresponding full trajectory from which a sub-trajectory, τ (Equation 2.37), is sampled. Here, L is the length of the full trajectory, S and A are the dimensions of the state and action spaces respectively, and d_e is the dimension of the PLE, which is a hyper-parameter. This mapping function f is designed to transform the high-dimensional trajectory data into a lower-dimensional preference embedding.

This mapping, f, should have the following properties: (1) it encodes similar trajectories into nearby points in the latent space; (2) it prevents leaking any information about τ from τ_{full}; (3) it handles τ_{full} of any horizon length; (4) it produces a normalized output range; (5) it allows for end-to-end learning with the decision diffuser. To achieve these properties, we design the mapping with the following sequence of operations:

1. We apply a fixed mask to τ_{full} to conceal information about τ. This step ensures that the PLE does not directly encode information about the specific sub-trajectory we're trying to predict, preventing information leakage.

2. We apply a learnable feed-forward layer which maps the state-action features into the PLE space. This layer allows the model to learn a suitable transformation of the input data.

3. We apply mean pooling to prevent time leakage and to handle variable horizon lengths. This operation allows the model to process trajectories of different lengths consistently.

4. We apply the sigmoid activation function for normalization, a choice motivated for adaptation described in the next subsection. This ensures that the output falls within a fixed range, facilitating stable training and adaptation.

5. Lastly, we compose each differentiable component to obtain the PLEs, z, which is then fed into the decision diffuser as context with the following objective:

$$\mathcal{L}_{\text{pretrain}}(\theta) = \mathbb{E}_{k\sim[1,K],x_0\sim q,\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\tau_k, f(\tau_{\text{full}}), k\right)\|^2\right]. \tag{5.6}$$

The proposed objective serves two crucial purposes in our approach: constructing a representation for the PLE placeholder that groups similar trajectories, and pretraining the model to generate trajectories adhering to the offline dataset's distribution. However, the model remains unaligned with specific user preferences at this stage. This general model serves as a starting point for further adaptation to individual preferences. The overall network architecture for pretraining is illustrated in the Figure 5.2 (left), providing a visual representation of how the various components interact during the pretraining phase.

5.4.2 Adaptation via preference inversion

We design a solution to quickly align the pretrained model to human preferences using a small human preference dataset. This approach aims to efficiently adapt the general model to specific user preferences without requiring extensive retraining. The basic idea is to learn a low-dimensional PLE, z, corresponding to preference labels, rather than performing full fine-tuning of the pretrained model. By focusing on learning only this compact representation, we anticipate faster convergence and improved stability in the adaptation process. We refer to the process of finding the matching PLE as *preference inversion*, drawing an analogy to the inversion methods for image manipulation described in Section 5.3. This terminology reflects the idea of "inverting" the model to find the preference embedding that best generates the desired trajectories.

We start the preference inversion process with a randomly initialized, learnable PLE, z. During pretraining, applying the sigmoid activation enables us to choose a prior bounded between 0 and 1. This bounded range provides a well-defined space for the preference embeddings, facilitating stable optimization and interpretation. To learn z, we freeze all weights of the diffuser model and backpropagate the loss gradients towards z. This approach effectively constructs trajectories that match human preferences while adhering to the data distribution sampled from the MDP during pretraining. By keeping the diffuser model fixed, we ensure that the learned preferences are consistent with the underlying dynamics and behaviors captured during pretraining.

To design a loss function that leverages pairwise preference labels, we sub-categorize the PLE into two types: winner PLE z_w, and loser PLE z_l. This categorization allows us to explicitly model the preference relationship between pairs of trajectories. By learning separate embeddings for preferred and non-preferred trajectories, we can capture the nuances of human preferences more effectively. This enables us to optimize and obtain an optimal z_w^* and z_l^* based on the reconstruction loss of the respective winner and loser trajectories, x^w and x^l as follows:

$$\mathcal{L}_{\text{inversion}}(z^w, z^l) = \mathbb{E}_{k\sim[1,K],x_0\sim q,\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}\left[\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta\left(\tau_k^w, z_w, k\right)\right\|^2+\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta\left(\tau_k^l, z_l, k\right)\right\|^2\right], \quad (5.7)$$

where θ is fixed. This loss function encourages the model to find preference embeddings that accurately reconstruct both the preferred and non-preferred trajectories, while maintaining the distinction between them.

A key advantage of having a frozen pretrained model is that it does not require the additional constraints to remain aligned to the base model, as is the case with Reinforcement Learning from Human Feedback (RLHF) or Direct Preference Optimization (DPO). This simplifies the adaptation process and potentially reduces the risk of instability due to over-deviation away from the base policy.

Furthermore, by performing partially fine-tuning using preference inversion, we may achieve comparable performance with less data and compute, similar to the success observed in parameter-efficient fine-tuning (PEFT) methods [189, 190]. This approach aligns with the growing trend in machine learning to develop more efficient adaptation techniques that reduce the computational and data requirements for personalization. In summary, the preference inversion method offers several potential benefits:

- Rapid adaptation: By focusing on learning a compact preference embedding, the model can quickly adjust to new preferences without extensive retraining.
- Data efficiency: The method potentially requires fewer preference labels to achieve good performance, making it more practical for real-world applications.
- Stability: By keeping the pretrained model fixed, we reduce the risk of overfitting or unstable behavior during adaptation.

5.4.3 Generating Preferred Trajectories

To sample a trajectory aligned with human preferences, we utilize a linear combination of the winner and loser PLEs, similar to the approach used in [78] to predict noise:

$$\begin{aligned} \hat{\boldsymbol{\epsilon}}_\theta\left(x_t, z_w^*, z_l^*, k\right) &= (1+v)\dot{\boldsymbol{\epsilon}}_\theta\left(x_t, z_w^*, z_l^*, k\right) - v\boldsymbol{\epsilon}_\theta\left(x_t, \varnothing, k\right), \\ \text{where } \dot{\boldsymbol{\epsilon}}_\theta\left(x_t, z_w^*, z_l^*, k\right) &= (1+u)\boldsymbol{\epsilon}_\theta\left(x_t, z_w^*, k\right) - u\boldsymbol{\epsilon}_\theta\left(x_t, z_l^*, k\right). \end{aligned} \tag{5.8}$$

Here, v and u are hyper-parameters. v controls the strength of the guidance, while u controls the influence of the loser information. To gain an intuition, we rewrite $\dot{\boldsymbol{\epsilon}}_\theta\left(x_t, z_w^*, z_l^*, k\right) = \boldsymbol{\epsilon}_\theta\left(x_t, z_w^*, k\right) + u(\boldsymbol{\epsilon}_\theta\left(x_t, z_w^*, k\right) - \boldsymbol{\epsilon}_\theta\left(x_t, z_l^*, k\right))$, which shows that we are pushing the score estimations away from the loser, originating at the winner. This allows us to efficiently leverage the pretrained model while quickly adapting to individual user preferences using a small amount of preference data. By learning only the low-dimensional PLE while keeping our pretrained model fixed, we reduce computational cost and enhance the stability of the adaptation process compared to fine-tuning the entire model. The overall proposed method is illustrated in the Figure 5.2.

5.5 Experiments

To rigorously assess the effectiveness of our method in integrating user preferences, we conduct a comprehensive evaluation addressing the following key questions:

1. **Preference Latent Embeddings:** Does the pretraining enable the PLE place-holder to be structured in a meaningful way, organizing similar trajectories close together?

2. **User Preference Alignment Efficiency:** Does our approach outperform baselines in aligning with user preferences when limited updates are available?

3. **Query Impact on Performance:** How does the number of preference queries influence the performance of each method?

4. **Human Preference:** Can the methods achieve similar performance when provided human-labeled, diverse and optimal queries?

5. **Hyperparameter Sensitivity:** How does the choice of prior and PLE dimension, d_e, impact the results?

6. **Fine-tuning Stability:** Does the fine-tuned model maintain its effectiveness with additional updates?

We conduct comprehensive evaluations to assess the effectiveness of our method in integrating user preferences. The evaluation addresses several key questions: the impact of the number of preference queries, the choice of prior, incorporation of z_l^*, and the stability of the adaptation process. Furthermore, we evaluate the capability to capture human preferences from diverse and optimal queries by creating our own custom preference dataset. To establish a strong benchmark, we compare our method against the following diverse baselines:

- **Diffuser:** A pretrained diffusion-based planner [191] representing the training data distribution but not adapted to user labels.
- **Guided Sampling:** Following RLHF, we train a reward model using the Bradley-Terry model [182], but employ classifier-guidance sampling [71], eliminating the need for further optimization with PPO [43].
- **Finetuning (Full):** This baseline directly fine-tunes the pretrained model using three different loss functions: DPO [166, 188], IPO [183], and a diffusion loss based solely on the winner's label. While all loss functions have been evaluated (details can be found in Section 5.5.2), Diffusion loss consistently performs the best and is used for this baseline.
- **Finetuning (LoRA):** This baseline provides a more direct comparison with our proposed method, where partial fine-tuning is performed. To achieve this, we utilize LoRA [190] with a rank of $r = 8$.

Hyper-parameters	Value
n_train_steps	1e6
n_diffusion_steps	20
horizon (halfcheetah)	4
horizon (others)	32
batch_size	32
learning_rate	2e-4
classifier_free_v	0.5

Table 5.1: **Settings for Diffuser and Decision-Diffuser**

Hyper-parameters	Value
n_train_steps	5000
n_diffusion_steps	20
horizon (halfcheetah)	4
horizon (others)	32
batch_size	32
learning_rate	2e-4
classifier_guided_v	0.1

Table 5.2: **Reward model settings (Guided Sampling)**

- **Preference Inversion (Proposed):** Our method partially fine-tunes the pretrained model to retrieve the optimal preference context. The pretraining stage utilizes a diffuser conditioned on masked trajectories.

Table 5.1 and 5.2 shows our hyper-parameter settings for our method and baselines respectively. For DPO and IPO, we set $\beta = 5000$. We also included a comparison with Preference Transformers [165] which does not use pretraining, in Section 5.5.4.

Experimental Setup. To examine the effectiveness of various personalization methods in automated decision-making systems, we tested our approach on a preference learning benchmark [165] that utilizes challenging control tasks from the d4rl dataset [1] in an offline setting. Specifically, we used the Hopper, HalfCheetah, and Walker2D tasks from the d4rl dataset, focusing on the medium-expert and medium-replay settings. Following the setup in Preference Transformers [165], we collected random query pairs without access to the task reward and assigned the winner to the query with the higher reward. To ensure a fair comparison, all baselines underwent 1 million updates during pretraining and $N_{\text{adapt}} = 5000$ updates during the alignment phase. Pretraining utilizes the full dataset for its respective tasks. During evaluation, we sample each method for 100 episodes across 5 different random seeds within their designated environments.

5.5.1 Latent Space Analysis

To understand the impact of our proposed mapping in the pretraining for the PLE placeholder, we conducted a visualization of the latent space representation. Using their respective pretrained models, we sampled 1000 random trajectories from each dataset and obtained the PLE,

z. These embeddings were then projected onto a two-dimensional space using t-SNE (with perplexity set to 30) as seen in Figure 5.3, with color intensity representing the normalized return of the corresponding masked trajectory. Examining the latent space of the medium-expert dataset, we observed distinct clusters representing low, medium, and high returns, respectively. These clusters were well separated and mostly linearly separable. In contrast, the medium-replay dataset, which consists of the entire replay buffer with a continual range of returns, exhibited a different pattern. The latent space reflected this continuous return distribution, showing not distinct clusters but rather a smooth transition of the latent embeddings based on their return values.

Overall, our proposed pretraining enables the PLE placeholder to be structured in an organized and meaningful manner, demonstrating that our proposed mapping, f, is able to organize similar trajectories close together. This organized latent space could accelerate the preference inversion process by first navigating the loss landscape to a local region of similar trajectories, and then refining the search for a more precise alignment with the true reward (human preference).

5.5.2 Preliminary Results: Finetuning with DPO and IPO

In our initial exploration of full-finetuning, we compared the performance of three loss functions: DPO [166, 188], IPO [183], and a diffusion loss focused solely on the winner's label. Figure 5.4 illustrates that both IPO and DPO loss exhibit instability across most tasks. Conversely, utilizing only the diffusion loss consistently matches or surpasses the performance of IPO and DPO in all scenarios.

5.5.3 Main Results

We compare our method against the baselines and assess the impact of the number of query pairs, N_{query}. We evaluate each model with N_{query} values of 10, 25, 50, and 100, analyzing their ability to align with user preferences using a small set of human-annotated queries and a limited number of updates ($N_{\text{adapt}} = 5000$). For the main experiment, we set $d_e = 16$ and $u = 0.02$.

Our experimental results are presented in Figure 5.5. In comparison to the other two baselines, our method consistently outperforms them, with the performance gap widening as the

number of queries decreases. This advantage is particularly pronounced in the hopper-medium-replay and walker2d-medium-replay tasks. Among the baselines, there is no clear overall performance winner. We observed that LoRA fine-tuning results are very close to full fine-tuning, with LoRA performing marginally better at lower query numbers (N=10 and N=25) and mixed results for N=50 and N=100. Preference Transformers performed the worst, likely due to its reliance on training from scratch using the learned reward model without leveraging any pretrained models. Besides, while the baseline methods consistently outperform the pretrained model (Diffuser) when $N_{\text{query}} > 50$, they often fail to achieve this with fewer queries, leading to negative adaptation. In contrast, our method manages to outperform the Diffuser in all cases. Remarkably, it exhibits exceptional robustness, maintaining a comparable score even when the number of queries N_{query} is drastically reduced to as small as 10 in most scenarios.

Overall, the results indicate a non-negative correlation between the score and N_{query}. For our proposed method, it is sufficient to use a minimum of $N_{\text{query}} = 50$ to retain the best performance for all tasks, except for the hopper-medium-expert task, which requires 100 queries. These findings validate our approach for resource efficient personalization, reducing the need for a large number of labeled data while requiring only a moderate number of updates.

5.5.4 Comparisons with Preference Transformers

We also compared our method with Preference Transformers (PT) [165], which is one of the state-of-the-art approaches for learning a preference reward for offline reinforcement learning. The key difference between our method and PT is that PT does not follow a pretrain-align framework and requires full training after the reward model is learned. Nonetheless, we conducted experiments to compare our method with PT, using full 1 million updates and 5000 updates. From Figure 5.6, PT completely fails when using only 5,000 updates since it does not leverage any pretrained model. Our method, using only 5,000 adaptation steps, is able to approach the same performance in medium-replay environments and matches PT in medium-expert environments.

5.5.5 Ablation Experiments

We perform a series of ablation experiments to gain deeper insights into the relative importance of different design choices and determine the sensitivity of our approach to variations in model components and hyperparameters.

Number of adaptation steps. Figure 5.7 shows that all methods peak around $N_{\text{adapt}} = 5000$. Preference inversion and guided sampling remain stable after minor drops at 15000 and 20000 updates, respectively. However, the performance of finetuning consistently declines after peaking, with rapid deterioration after $N_{\text{adapt}} = 30000$, suggesting excessive deviation from the base model. The stability of our method is advantageous for real-world deployment, where the optimal stopping point is often unknown in advance.

Loser PLE. Figure 5.8 shows that incorporating the loser PLE, z_l^* with $u > 0$, consistently improves the sampling performance compared to using only the winner PLE with $u = 0$. The improvements peak at $u = 0.02$, resulting in $1 \sim 3\%$ gains across various N_{query} values, and gradually diminish as u decreases further. Utilizing z_l^* provides a small boost when N_{query} is high, but notably enhances the sampling when N_{query} is low.

Choice of prior and PLE dimension. Given that our PLE z is constrained to the range [0, 1], we test three different priors for initialising z within this interval: Uniform$[0, 1]$, Gaussian$(0.5, 0.5/3)$, and Fixed_0.5. Figure 5.9 demonstrates that all three settings perform comparably well, with the uniform prior slightly outperforming the others. Similarly, varying the PLE dimension d_e across 2, 4, 8, 16, and 32 consistently yields good results, demonstrating relative insensitivity to this hyper-parameter, with a slight advantage observed at $d_e = 16$.

Recommendations. Our ablation studies reveal that the best performing configuration is to utilize $d_e = 16$ for pretraining, followed by initializing z using a uniform prior during adaptation with $N_{\text{adapt}} \geq 5000$ steps, and incorporating the loser PLE, z_l^* for sampling.

5.5.6 Real Human Preference on Quality Diversity Dataset

Previous experiments, following the design of Preference Transformers [165], focus on recovering a hidden task reward. While useful for validating human preferences, it prioritizes optimality over diversity, and may not fully capture real-world scenarios. In contrast, practical decision-making often involves selection from a **diverse** set of **high-reward** trajectories. To better reflect this, we design a new experiment based on *Quality Diversity* (QD) [192–194]. In policy learning, QD refers to an algorithm's ability to discover diverse, high-performing policies with distinct behaviors.

To implement this, we train a set of QD policies as described in [194], generating a diverse dataset of 750 optimal Walker2D episodes for model pretraining without preference alignment. We collected our own preference dataset using the following process:

- **Participants**: Three users with no prior robotics expertise were recruited to minimize bias.
- **Stimuli**: Each participant evaluated 100 randomly generated pairs of video animations showcasing diverse Walker2D gaits (e.g., varying speeds, strides, and styles), produced using QD policies.
- **Task**: Participants selected their preferred motion based on intuitive criteria such as smoothness, stability, and naturalness. They were instructed to maintain a consistent selection strategy and document their decision criteria in writing.
- **Output**: Each comparison resulted in a winner (preferred) and loser (less preferred) label, used for training.

We then adapt the pretrained policy to each user's preference labels, consisting of 100 query pairs each. For evaluation, we generate 100 trajectories per baseline and ask users to choose the closest match to their preference criteria. The survey results, shown in Figure 5.11, indicate that our proposed method receives the vast majority of votes, demonstrating its effectiveness in capturing human preferences. Figure 5.10 displays the sampled trajectories from the aligned model, generated using preference inversion, which closely match the users' descriptions.

5.6 Conclusion

This work presents a novel approach that enables a policy to quickly adapt to a small human preference dataset. It consists of pretraining followed by adaptation on latent embeddings via preference inversion for rapid alignment. Evaluation results demonstrate that our method adapts more accurately to human preferences with minimal preference labels, outperforming baselines in both offline datasets and our custom dataset with real human labels. This promising method shows potential for further applications across diverse settings.

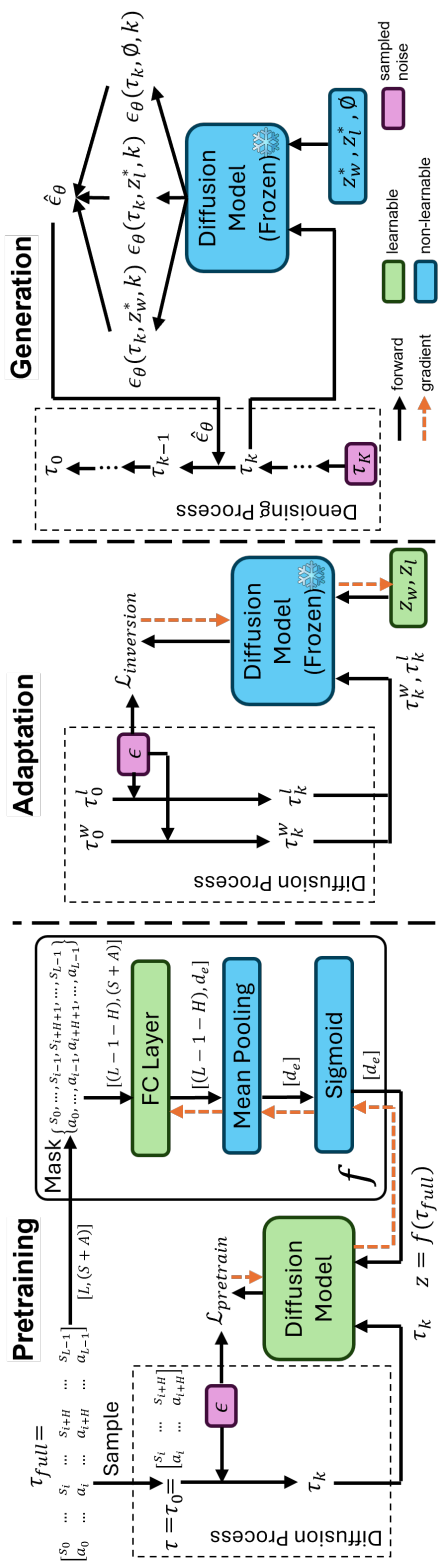


Figure 5.2: **Overview of the proposed method.** (Left) Pretraining: A placeholder for preference latent embedding (PLE), z, is co-trained with the diffusion model, without reward supervision. (Middle) Adaptation: With diffusion model weights frozen, PLEs are aligned to user labelled query pairs via preference inversion. (Right) Generation: Conditional sampling with optimal PLEs generate trajectories that match the users' preference.

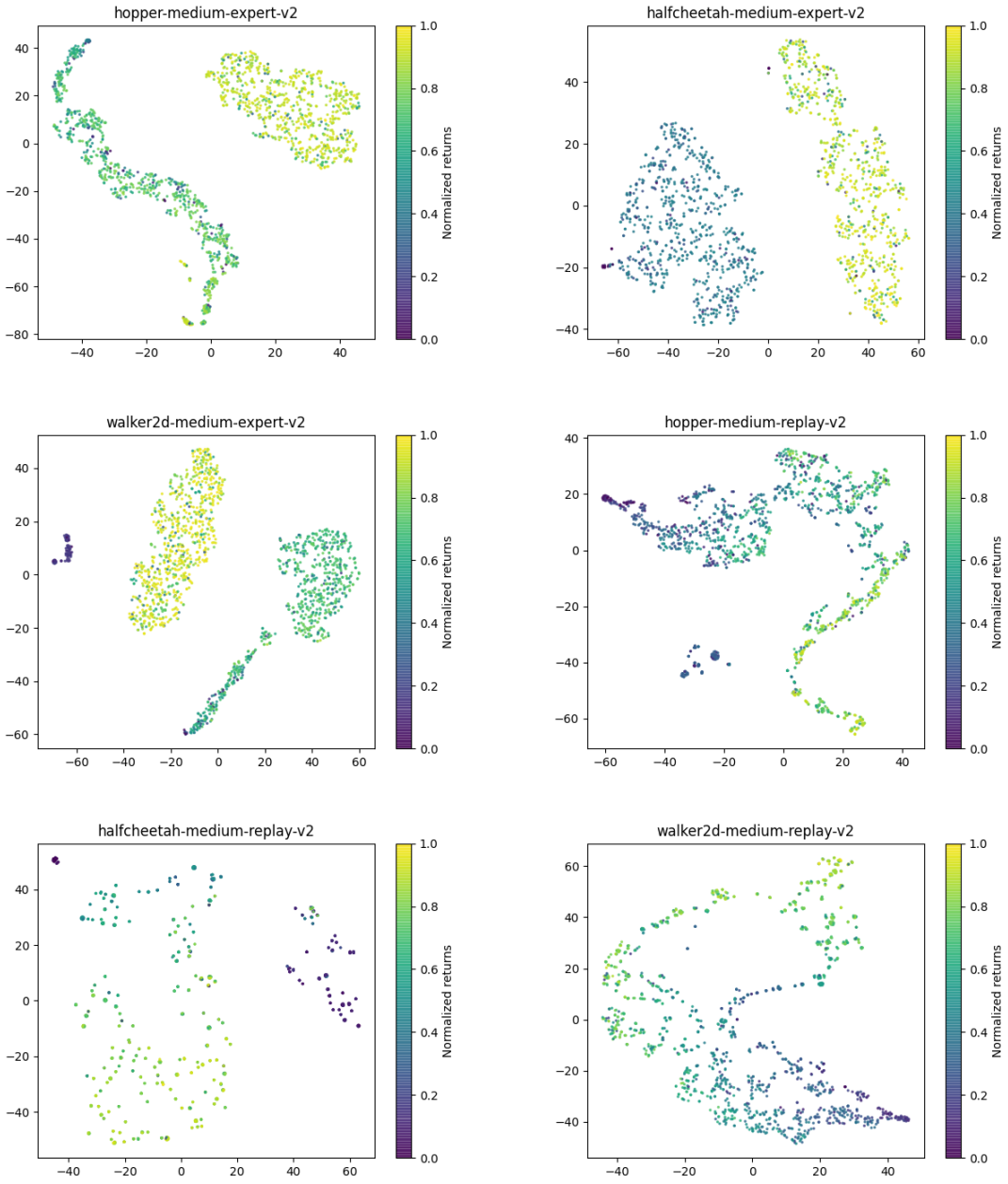


Figure 5.3: **Latent space analysis**: We visualize t-SNE plots of PLEs post-pretraining. Each point represents a trajectory in PLE space, with color intensity indicating its normalized return.

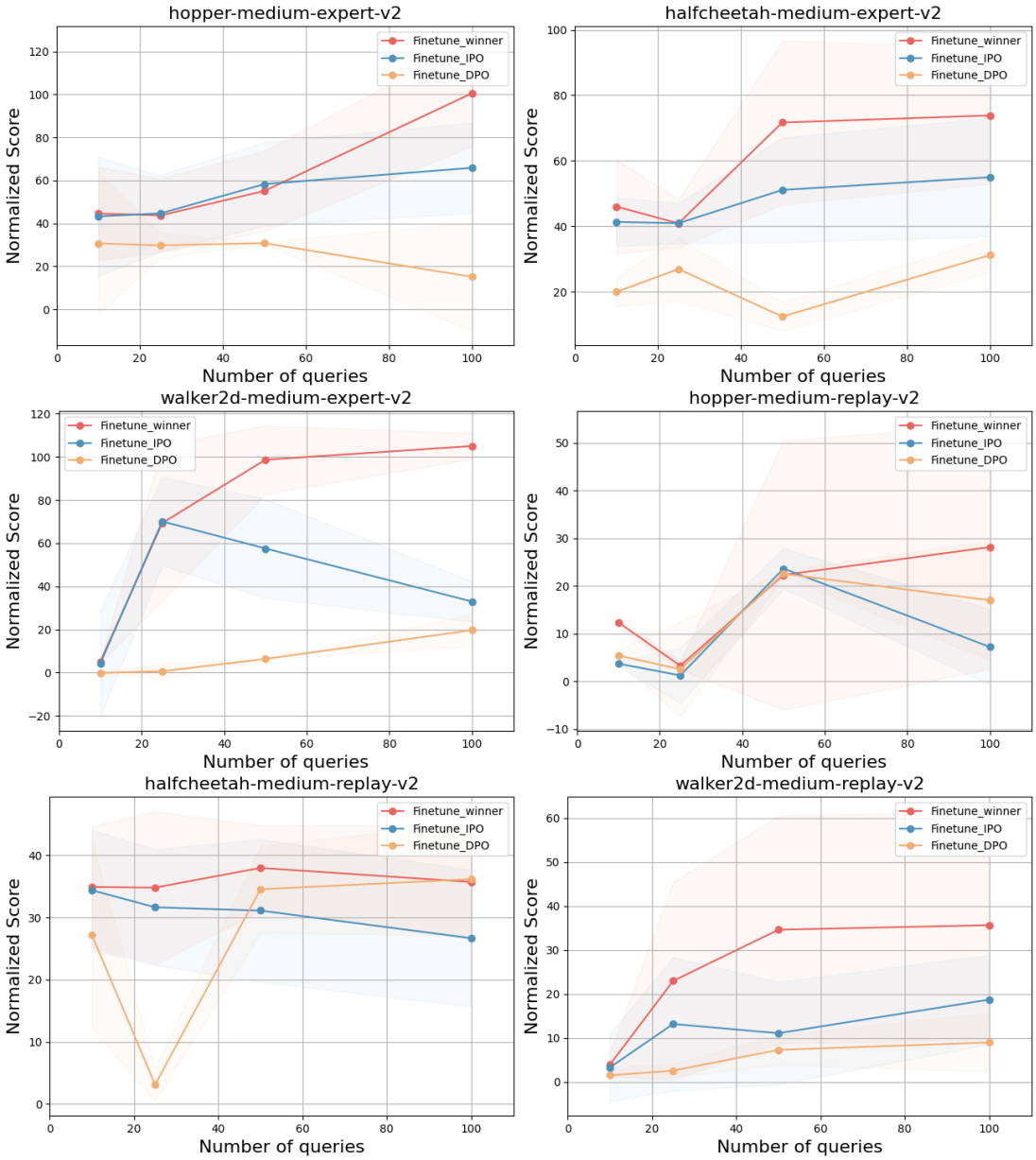


Figure 5.4: **Comparisons between different losses for finetuning baseline.**

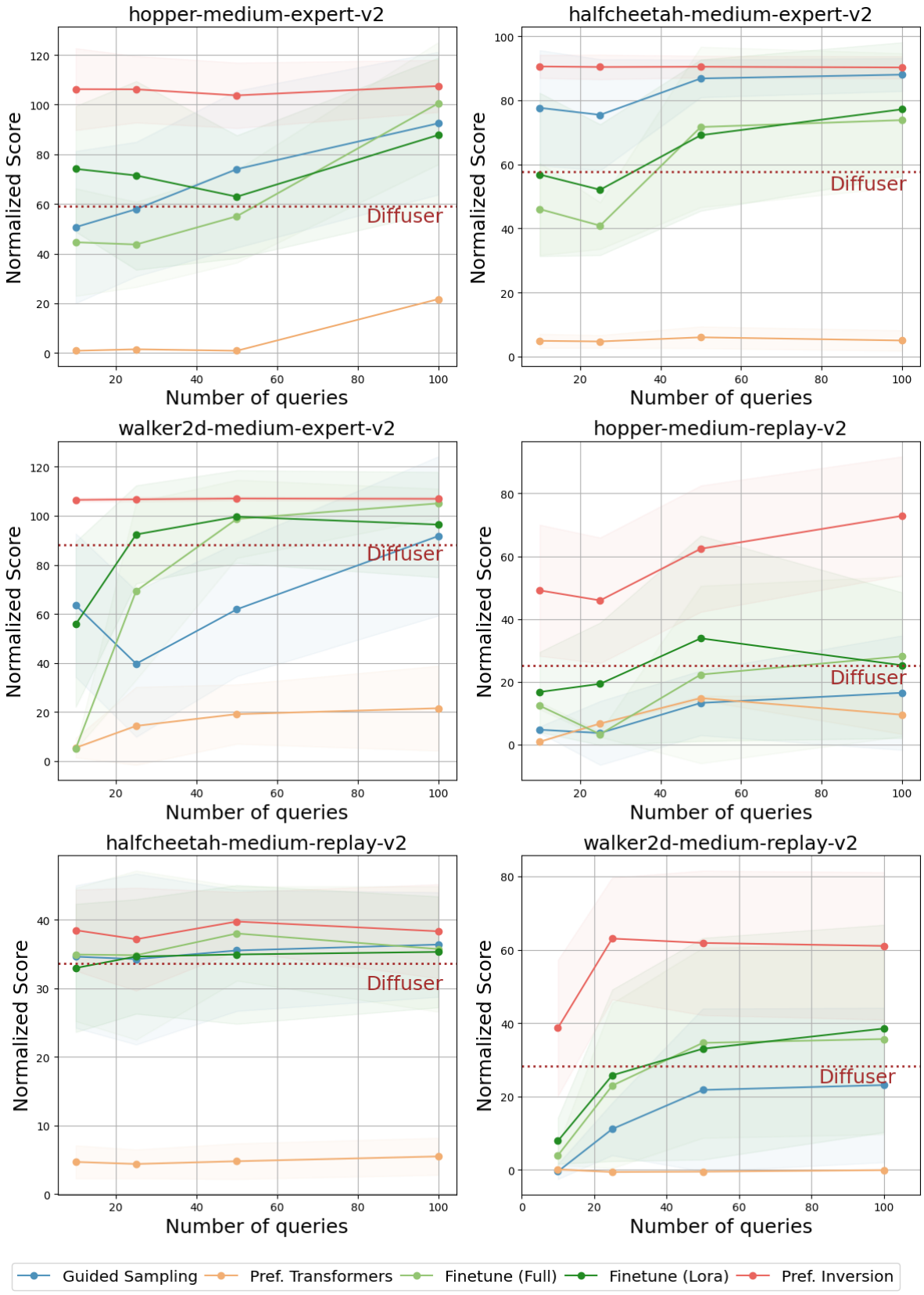


Figure 5.5: **Main results evaluated over different numbers of queries across six control tasks.** We report the normalized score as defined in [1]

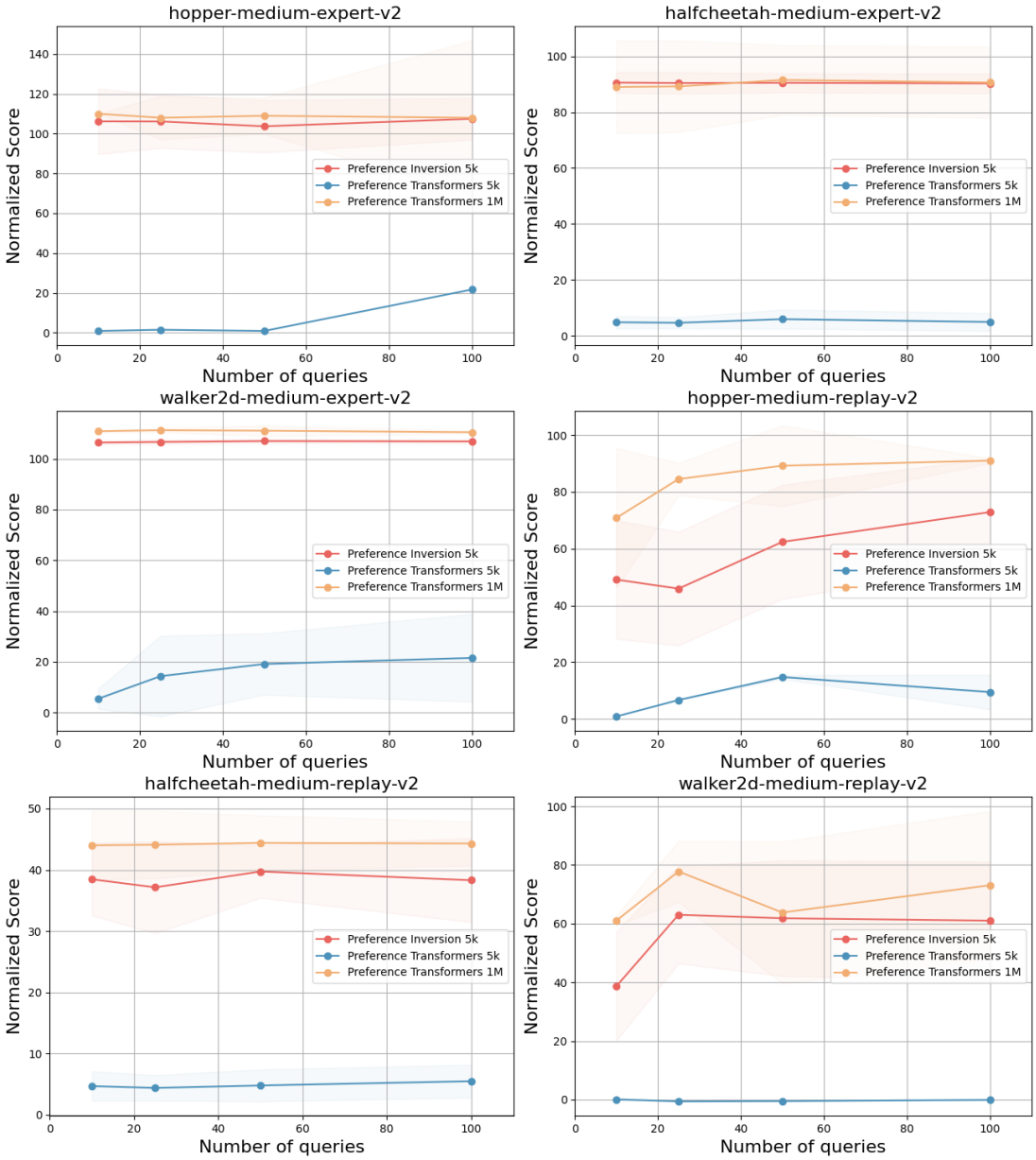


Figure 5.6: **Comparison with Preference Transformers**

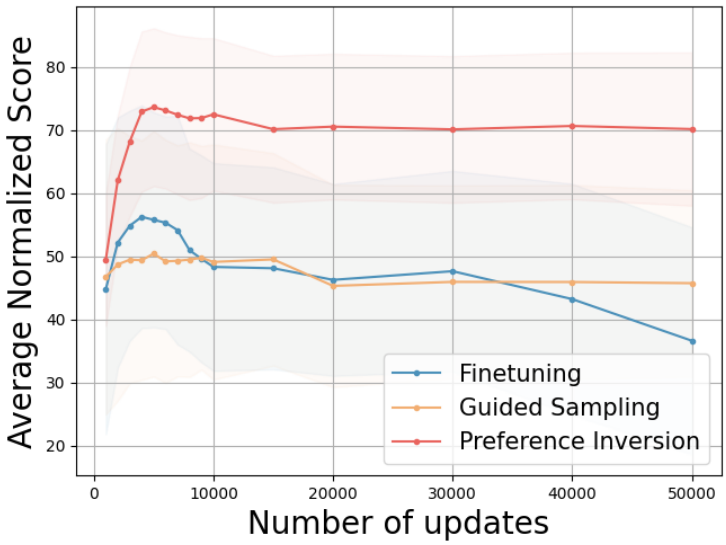


Figure 5.7: **Ablation: Adaptation stability across N_{adapt}.**

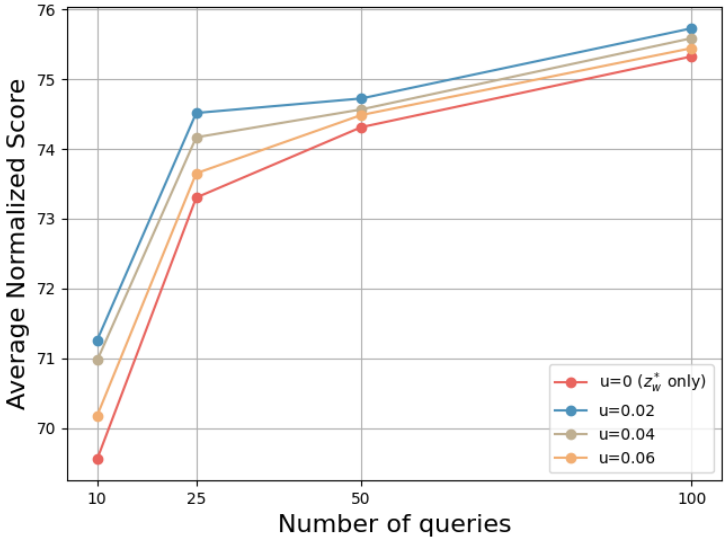


Figure 5.8: **Ablation: The impact of utilizing loser PLE, z_l^* for sampling**

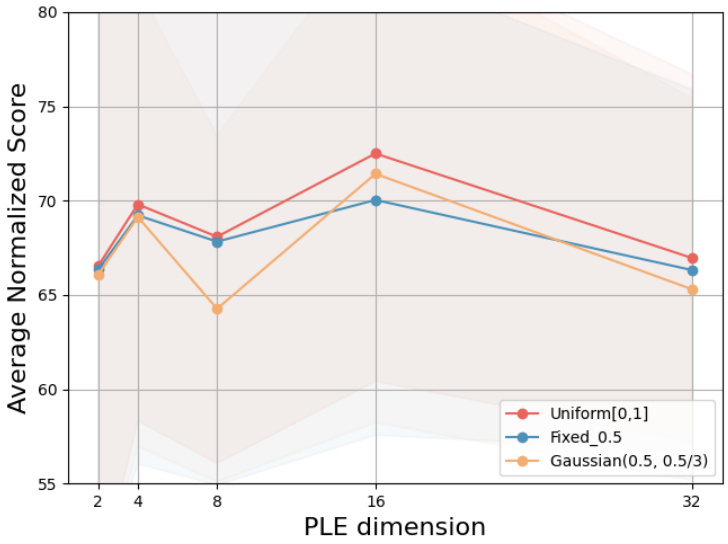
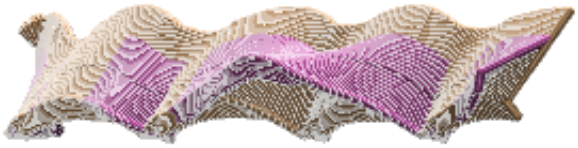
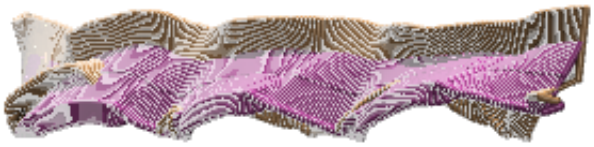


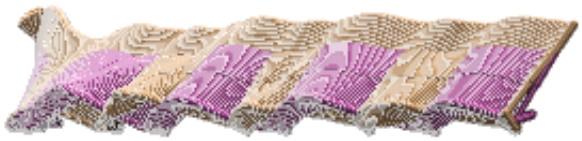
Figure 5.9: **Ablation: Choice of different priors for initialization and PLE dimension**



(a) User A: 'Back leg swing up high and body leaning forward'



(b) User B: 'Body upright, front knee bent, rear leg 45degrees'



(c) User C: 'Gentle hopping with moderately fast strides'

Figure 5.10: **Trajectories generated using our proposed method, an aligned model conditioned on user's respective PLE.** The samples closely match each user's description of their preference.

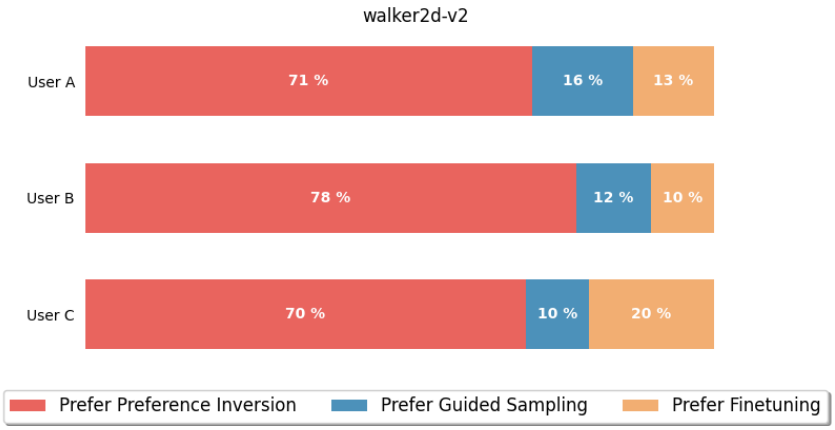


Figure 5.11: **Preference Learning Survey Results.** Users select their preferred trajectories from samples generated by both baseline and our proposed method.

Chapter 6

Generalization of Human Pedestrians Behaviors

6.1 Introduction

Robot navigation in crowded environments is a critical challenge in robotics with significant real-world implications. As robots become increasingly integrated into our daily lives, their ability to safely and efficiently maneuver through spaces populated by human pedestrians becomes essential. This capability is essential for various applications, including autonomous delivery services in urban areas, assistive robots in healthcare settings, and mobile robots in bustling industrial environments [5]. Effective crowd navigation enables robots to perform tasks in public spaces without causing disruptions or safety hazards, enhancing their utility and acceptance in society [195]. Moreover, robots that can navigate crowded environments can potentially improve emergency response scenarios, assist in crowd management during large events, and contribute to the development of smarter, more efficient urban spaces. By mastering this skill, robots can seamlessly operate alongside humans, leading to more harmonious human-robot interactions and paving the way for broader adoption of robotic technologies in our increasingly populated and dynamic world.

For RL-based motion planners, the environment in which the agent learns plays a pivotal role in shaping its decision-making capabilities. The environment serves as the agent's training ground, providing it with the necessary experiences to develop an optimal policy for navigating and interacting with the world [15, 17, 18, 34, 196–200]. In the context of motion planning, a crucial aspect of the environment is its ability to accurately represent the inherent diversity and

unpredictability of pedestrian movements. Pedestrians exhibit a wide range of behaviors, influenced by factors such as individual preferences, social norms, and environmental conditions. They may walk at different speeds, change directions abruptly, engage in conversations, or exhibit other unique movements. To enable an RL agent to learn a robust and adaptable policy, the environment must comprehensively capture this spectrum of pedestrian behaviors. Failure to adequately model the diversity of human movements can restrict the agent's ability to generalize to unseen scenarios, making it difficult to navigate effectively and safely in real-world crowded spaces.

Numerous approaches have been proposed to generate pedestrian movements for training RL-based local motion planning policies. These approaches can be broadly categorized into two distinct groups, each with its own set of limitations. The first category involves single-agent approaches. One straightforward method involves assigning waypoints to pedestrians from a dataset [21]. However, this approach does not capture the complex interactions between robots and pedestrians, as they do not influence each other's behaviors. To address this limitation, some studies have manually designed pedestrian behaviors based on crowd density [201, 202] or employed fixed non-learning-based algorithms to control pedestrians [18, 23, 63, 203]. While these methods attempt to introduce more realistic pedestrian movements, they may still suffer from limited diversity and potentially lead to overfitting.

The second category of approaches for generating pedestrian movements in RL-based local motion planning reframes the problem as a decentralized multi-agent collision avoidance task [15, 17, 19]. In this paradigm, each agent (pedestrian or robot) operates independently, learning to navigate towards its designated goal while concurrently avoiding collisions with other agents in the environment. This learning process occurs dynamically during training, enabling the agents to adapt their behaviors in response to the actions of others.

This decentralized multi-agent approach presents two key advantages. Firstly, it eliminates the need for explicit specification of each pedestrian's behavior, thereby avoiding potential biases introduced by manual design choices. This allows for a more naturalistic and flexible representation of pedestrian movements, as the agents learn to navigate and interact based on their own experiences and observations within the environment. Secondly, this approach exhibits high sample efficiency due to policy bootstrapping. In this context, bootstrapping refers to the ability of each agent's trajectory to contribute to the learning of the shared policy

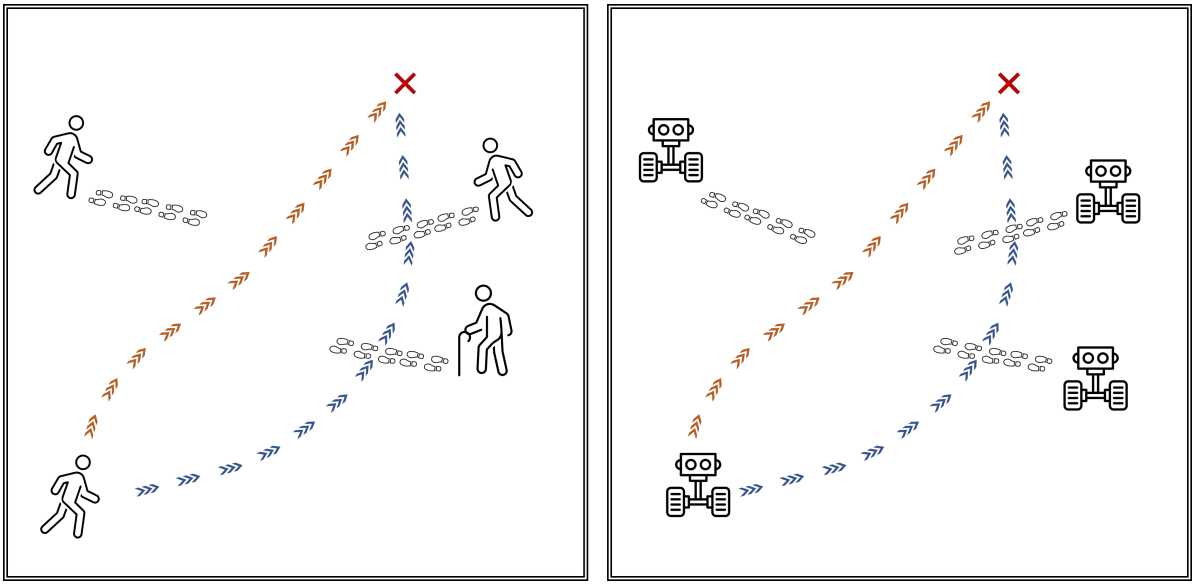


Figure 6.1: **Overview of diversity framework.** A human may take diverse strategies to reach the same predefined goal (left). We propose a behavior-conditioned policy to integrate such diversity into the robot agent (right). This diversity enriches the agent with a more varied range of experiences when learning in a multi-agent framework, and improves its ability to generalize in unseen crowd behaviors.

model, effectively increasing the amount of training data available and accelerating the learning process. However, despite these advantages, decentralized multi-agent approaches face practical challenges when deployed in real-world scenarios characterized by diverse or unforeseen dynamics. The learned policy, often trained on a limited set of behaviors, tends to assume homogeneous behaviors among multiple agents upon deployment which may not hold true in general. Consequently, the robot may struggle to generalize well and navigate effectively and safely in scenarios with varied crowd behaviors.

In this chapter, we propose a novel sample-efficient multi-agent framework to enhance behavior diversity among agents. Diverse movements induced by different agents enrich experiences and enhance robustness to unpredictable behaviors in unseen or challenging scenarios. Our framework introduces the concept of behaviors conditioned on a policy. These behaviors, randomly sampled as token embeddings by each agent at the start of each training episode, incentivize agent diversity. We assign intrinsic rewards for agents to take varied actions for every state conditioned on the sampled behavior. These rewards are based on a discriminator capable of identifying behavior from a state-action pair. With this approach, we can train robust RL policies for local motion planning in highly complex environments.

6.2 Related Work

6.2.1 Pedestrian Modelling

In the context of RL-based local motion planning, the movement of dynamic obstacles, often represented by human pedestrians, is a crucial environmental factor. The behavior of each pedestrian significantly influences both the overall dynamics of the environment and the learning process of the robot agent within the RL framework. Existing approaches to modeling pedestrians in this context can be broadly classified into two following categories:

Single-agent approaches. In these approaches, only a single robot agent learns to navigate within the crowd, while the behavior of pedestrians is typically modeled using non-learning-based algorithms. One common strategy is based on the velocity obstacles (VO) [204], which computes a region of collision under the assumption that other robots maintain their current velocities. Some examples VO-based pedestrian modelling can be found in [18, 23, 63, 203] Another strategy is based on the social forces [201, 202], which take into account the acceleration toward the desired velocity while keeping a certain distance to other pedestrians. Lastly, handcrafted ideas have also been proposed to model pedestrians' behaviors. For example, Wang et al. [205] proposed the simulated agent dynamics using a second-order physical system parameterized by force, mass, and friction values. However, one common drawback to these approaches is that the RL-agent might overfit to the chosen pedestrian behaviors during training. Furthermore, due to the non-learning nature of the pedestrians in these approaches, only the robot agent undergoes learning, leading to inefficient utilization of experiences and lower sample efficiency during training.

Multi-agent approaches. The core principle of the multi-agent framework in the context of simulating pedestrian behavior and training robot navigation policies is to control multiple agents within the environment using a unified policy [15–17, 19, 22, 24, 206, 207]. Each agent, be it a pedestrian or the robot itself, operates under the guidance of this shared policy, which it learns through interactions and observations within the environment. This multi-agent setup naturally leads to the emergence of dynamic movements during the training process. As each agent strives to achieve its goal while adhering to the constraints of collision avoidance, it must

continuously adapt its actions in response to the movements of other agents. This results in a complex and interactive dance of movements within the environment, simulating the dynamic nature of real-world crowded scenarios. The shared policy, updated based on the experiences of all agents, evolves to capture the nuances of these interactions, ultimately guiding the agents towards efficient and collision-free navigation strategies.

However, a potential drawback of this multi-agent framework is the tendency for agents to converge towards homogeneous behaviors over time. This behavioral convergence can be observed in our preliminary training using this framework as demonstrated in the accompanying video (https://youtu.be/EevMn2-ZNng). Since all agents share the same policy and learn through a bootstrapping process, where each agent's experience contributes to the overall learning, their individual behaviors may gradually align and become increasingly similar. While this convergence can be beneficial in terms of coordination and cooperation, it may also limit the diversity of behaviors exhibited by the agents. This limitation can be problematic in scenarios where the real-world environment presents a wide range of pedestrian behaviors.

6.2.2 Behavior Diversity in RL

To tackle the issue of homogeneity in agent behaviors, a range of approaches has been explored to promote diversity. In single-agent scenarios, one prevalent solution involves maximizing the entropy of the policy alongside the reward objective [208]. This encourages the agent to explore a wider range of actions and discover diverse strategies for achieving the same goal, thus preventing the policy from converging to a single, deterministic solution. However, this entropy based diversity is specific for the actions of a single agent and does not increase the diversity among multiple agents. Eysenbach et al. [159] introduced Diversity is All You Need (DIYAN), a method that further enhances agent behavior diversity by maximizing the mutual information between skills and states. This approach promotes more comprehensive exploration of the state space, enabling the discovery of a wider variety of potential solutions and improving the agent's adaptability to novel situations.

In the multi-agent domain, efforts to increase behavior diversity have focused on the Centralized Training with Decentralized Execution (CTDE) framework [209–211]. CTDE involves training agents with a centralized critic that possesses global information, while the execution of the learned policies remains decentralized, with each agent acting independently based on its

local observations. When agents are tasked with different objectives within the CTDE framework, they are often assigned distinct policies, each specialized for its specific task, while sharing a common critic network for evaluating the overall performance. This multi-policy approach naturally fosters diversity in agent behaviors as each policy is tailored to a particular objective. To further enhance this diversity, several strategies have been proposed, including encouraging social interactions among agents, rewarding individualistic behaviors, and promoting the exploration of diverse strategies [97, 212, 213]. However, one potential drawback of multi-policy approaches is a reduction in sample efficiency during training. This is because each agent primarily updates its own policy based on its individual experiences, rather than contributing to the learning of a unified policy shared by all agents. As a result, the learning process may require more data or interactions to achieve comparable performance compared to approaches that utilize a single shared policy.

6.3 Approach

We present our approach to learning robust agents through behavior diversity. Instead of using multiple policies to create diversity as in CTDE, we opt for a more sample-efficient method by using only a single policy. We first formulate the agent behaviors, and how they can be used to generate diversity among agents (Section 6.3.1). Then we explain how the behaviors and diversity can be integrated together within a single policy (Section 6.3.2). Finally, we describe how to train a behavior-conditioned policy in an end-to-end manner with all the integrated components (Section 6.3.3).

6.3.1 Agent Behavior

In Figure 6.1, people's approaches to walking towards a goal can vary: some prioritize speed with a longer path, while others choose a shorter route at a slower pace. When avoiding moving obstacles, some turn left, while others turn right. Although individuals may have unique behaviors, there can be similar patterns. We formalize this with discrete behavior tokens $h \in [0, 1, \ldots, M-1]$, where M is the total number of distinct behaviors. Each token represents a distinct pedestrian behavior, and different agents may share the same token.

To foster diversity amongst different agents, our goal is to assign different behavior tokens to different agents to exhibit distinct behaviors. In other words, for every state s, agents should

perform different actions a depending on the assigned h. More formally, this idea can be formalised using information theory by maximizing the mutual information $I((S,A);Z)$, where (S,A) is the joint distribution of S and A. $Z \sim p(h)$, S, and A represent the random variables for behavior, state, and action respectively. Additionally, the diverse actions performed for different h should arise for every state instead of exploiting only certain states. For this, we minimize $I(S,Z)$ as a regularizer. In sum, we maximize

$$\begin{aligned}\mathcal{F}(\theta) &\triangleq I((S,A);Z) - I(S;Z) \\ &= (H[Z] - H[Z \mid S,A]) - (H[Z] - H[Z \mid S]) \\ &= -H[Z \mid S,A] + H[Z \mid S],\end{aligned} \tag{6.1}$$

where H is the Shannon entropy. The first term implies it is easy to infer the behavior h given any (s,a). This makes sense intuitively as it means the agents are distinguishable due to their diverse behaviors and not behaving in a homogeneous way. The second term implies that the agents' behavior should not be distinguishable exclusively given s. It is intractable to compute $p(h|s)$ and $p(h|(s,a))$ by integrating all states, actions, and skills. So we approximate the posteriors with learned discriminators $q_\phi(h|s)$ and $q_\psi(h|(s,a))$. We instead optimize the variational lower bound derived using Jensen's Inequality [214]:

$$\begin{aligned}\mathcal{F}(\theta) &= -H[Z \mid S,A] + H[Z \mid S] \\ &= \mathbb{E}_{z\sim p(h), s\sim\pi(h)}[\log p(h \mid s)] \\ &\quad - \mathbb{E}_{z\sim p(h), s\sim\pi(h), a\sim\pi(s,h)}[\log p(h \mid s,a)] \\ &\geq \mathbb{E}_{z\sim p(h), s\sim\pi(h)}[\log q_\phi(h \mid s)] \\ &\quad - \mathbb{E}_{z\sim p(h), s\sim\pi(h), a\sim\pi(s,h)}[\log q_\psi(h \mid s,a)] \\ &\triangleq \mathbb{G}(\theta),\end{aligned} \tag{6.2}$$

where $s \sim \pi(h)$ means to first sample the action a from π followed by sampling the environment to get the state s. It is non-trivial to directly optimize θ via maximizing the lower bound $\mathbb{G}(\theta)$ since $s \sim \pi(h)$ has to be sampled through a non-differentiable simulator. Below we introduce how to optimize θ using an intrinsic reward alongside the RL objective.

6.3.2 Behavior-Conditioned Policy

First, we incorporate the idea of behaviors into our policy where we condition our policy on the agent's behaviors. Each agent, i, sample their actions from a shared behavior-conditioned

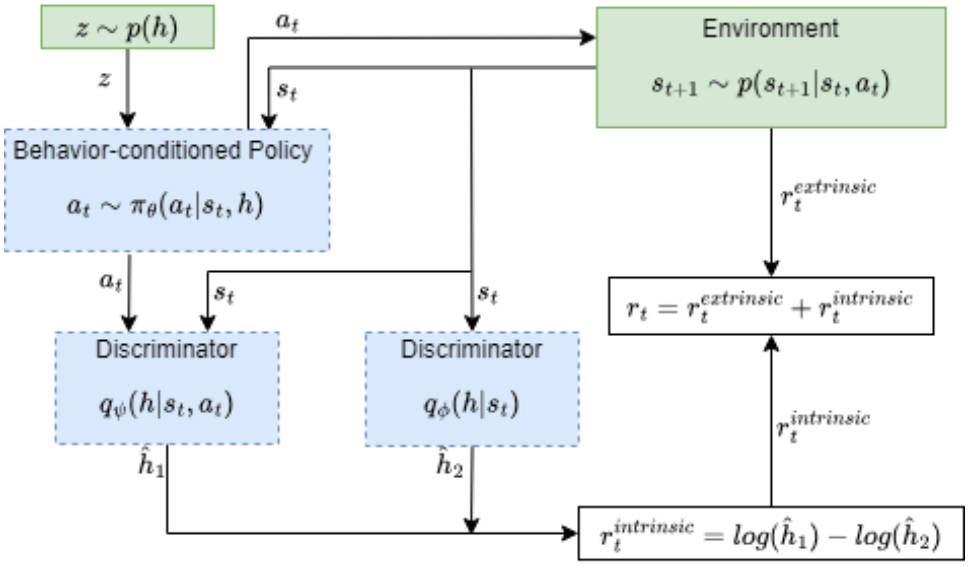


Figure 6.2: **Our framework for behavior-conditioned policy**. An intrinsic reward is computed based on discriminators q_ψ and q_ϕ, which encourages the diversity by indirectly maximizing the lower variation bound $\mathbb{G}(\theta)$

policy as $a \sim \pi_\theta(\cdot, s_t^i|h^i)$, for behavior token ID h^i at timestep t. Each behavior token maps to an embedding in the policy network, enabling the policy to generate distinct behaviors for agents. To maximize such diversity, we introduce an intrinsic pseudo-reward r_{int} motivated from maximizing $\mathbb{G}(\theta)$ derived previously:

$$r_t^{int} = log[q_{\psi_{sa}}(h \mid s_t, a_t)] - log[q_{\psi_s}(h \mid s_t)]. \tag{6.3}$$

Maximizing the intrinsic pseudo-reward through reinforcement learning allows maximizing $\mathbb{G}(\theta)$ despite sampling $s \sim \pi(h)$ from a non-differentiable simulator. In Eqn. (6.3), a_t is sampled from a policy conditioned on behavior rather than a default policy, as generating diverse actions requires knowledge about h. The proposed intrinsic reward promotes action diversity while learning the main task. Overall, our intrinsic reward shares some similarity to DIAYN [159] in which both use token-conditioned policies. However, the difference is that our method encourages agents with different tokens to generate diverse actions for a given state instead visiting diverse states. Figure 6.2 shows an overview of the interaction between the behavior-conditioned policy and the discriminators.

Algorithm 3 Behavior-conditioned policy for N agents

Initialize policy network π_θ, value function V_ϕ, discriminators $q_{\psi_{sa}}$ and q_{ψ_s}
Require: hyper-parameters $\alpha, \gamma, \lambda, \epsilon, M$
while not converged **do**
  // Collect data in parallel
  for $i = 1, 2, \ldots,$ N **do**
    Sample behavior $h^i \sim p_M(h)$
    Sample behavior-conditioned policy $\pi_\theta(\cdot, s_t|h)$ for T_i timesteps, collecting $\{s^i_{t+1}, a^i_t, r^i_t\}$ where $t \in [0, T_i]$
    Modify reward by adding bonus intrinsic reward $r^i_t \leftarrow r^i_t + \alpha\left\{log[q_{\psi_{sa}}(h \mid (s^i_t, a^i_t)]) - log[q_{\psi_s}(h \mid s^i_t)]\right\}$
    Compute advantages using GAE [215] $\hat{A}^i_t = \sum_{l=0}^{T_i}(\gamma\lambda)^l(r^i_t + \gamma V_\phi\left(s^{t+1}_i\right) - V_\phi\left(s^i_t\right))$
  end for
  $\pi_{old} \leftarrow \pi_\theta$
  // Update Policy, Value Functions and Discriminators
  for $j = 1$ to $epoch_\pi$ **do**
    Compute Ratio $k_t = \frac{\pi_\theta(a^i_t|o^i_t)}{\pi_{\text{old}}(a^i_t|o^i_t)}$
    $\mathbb{L}^{PPO\_clip}(\theta) = \sum_{t=1}^{T_{\max}} min\left(k_t\hat{A}^i_t, \text{clip}\left(k_t, 1-\epsilon, 1+\epsilon\right)\hat{A}^i_t\right)$
    Update θ using Adam w.r.t. $\mathbb{L}^{PPO\_clip}(\theta)$
  end for
  for $j = 1$ to epoch_v **do**
    $\mathbb{L}^V(\phi) = -\sum_{i=1}^{N}\sum_{t=1}^{T_i}\left(\sum_{t'>t}\gamma^{t'-t}r^i_{t'} - V_\phi\left(s^i_t\right)\right)^2$
    Update ϕ using Adam w.r.t. $\mathbb{L}^V(\phi)$
  end for
  for $j = 1$ to epoch_d **do**
    $\mathbb{L}^D(\psi_{sa}) = -\sum_{i=1}^{N}\sum_{t=1}^{T_i}\left(h^i \cdot \log\left(q_{\psi_{sa}}(s^i_t, a^i_t)\right)\right)$
    $\mathbb{L}^D(\psi_s) = -\sum_{i=1}^{N}\sum_{t=1}^{T_i}\left(h^i \cdot \log\left(q_{\psi_s}(s^i_t)\right)\right)$
    Update ψ_{sa}, ψ_s using Adam w.r.t. $\mathbb{L}^D(\psi_{sa}), \mathbb{L}^D(\psi_s)$
  end for
end while

6.3.3 Training Procedure

To encourage diverse agent behaviors, we shift from the diffusion planner framework used in earlier chapters, which focuses on offline datasets. While offline data can capture trajectory variations, it falls short in multi-agent settings: these datasets may show differences in paths but fail to reflect meaningful distinctions in how agents act or make decisions. This occurs because pre-recorded data lacks real-time interactions where agents adapt to other agents' evolving behaviors, a process essential for fostering diversity. To address this, we train agents online, enabling continuous interaction with the environment and other agents. Through this approach, agents gradually develop distinct strategies, creating a cycle where diversity increases naturally as they learn. Such adaptive behavior cannot be replicated with offline data alone.

We build upon the training procedure from [17], alternating between sampling trajectories and updating the policy via the PPO algorithm [43]. Each agent uses an identical policy to collect data until a batch is gathered. Algorithm 3 outlines the training details. Key differences from [17] are highlighted in blue:

1. At each episode start, agent i samples a new behavior token $h^i \sim p_M(h)$, with $p_M(h)$ being a discrete uniform distribution with M behaviors (Line 6). This token is mapped to a 32-dimensional continuous embedding.

2. Agents sample from a policy conditioned on h^i (Line 7), allowing for varied actions based on behavior. This behavior-conditioned policy enables agents to take different actions depending on the sampled behavior.

3. Intrinsic rewards for each agent are computed using discriminators $q_{\psi_{sa}}$ and q_{ψ_s}, parameterized by ψ_{sa} and ψ_s respectively, based on Eqn. (6.3). These rewards are added to the task reward in the replay buffer (Line 8).

4. We optimize discriminators $q_{\psi_{sa}}$ and q_{ψ_s} with cross-entropy loss (Line 25) using the Adam optimizer [216]. Adding one standard deviation of Gaussian noise to discriminator inputs helps prevent overfitting. The loss is computed between predicted behavior tokens $\hat{z}$ from on-policy samples and ground truth behavior.

6.4 Experiments

We conduct comprehensive experiments to demonstrate the effectiveness of our method over previous solutions. We simulate these experiments with new crowd behaviors not encountered during agent training.

6.4.1 Experimental Setup

Implementation. We simulate a large-scale group of robots using Stage [217], a popular robot simulator widely used in multi-agent research. Each agent is initialized as a non-holonomic differential drive robot ($0.5m \times 0.5m$) equipped with a 2D-laser scanner to sense its surroundings. The 2D laser is set to 360 degrees FOV with a max range of 10m. For baselines and our proposed method, we follow the same setup as [17] in terms of task states and reward formulation, PPO loss and neural network backbone to process the sensor data. We make one change to the NN backbone by adding a behavior embedding derived from the behavior token, h, for the neural network input. Each discriminator is modeled with a two-layer feed-forward network with 128 hidden units and ReLU activations [218]. Table 6.1 lists the hyper-parameter settings.

Hyper-parameter	**Value**
Discount Factor γ	0.99
PPO Smoothing λ	0.95
PPO Clip Value ϵ	0.1
# Epoch for Policy Network	3
# Epoch For Value Network	3
# Epoch for Discriminators	1
Advantange Weight α	0.1
PPO Learning Rates	0.00005
Discriminators Learning Rates	0.00005
Epoch_π, Epoch_d, Epoch_v	3

Table 6.1: **Hyper-parameters in our implementation.**

Training Setup. We train agents in a realistic, heavily trafficked $20m \times 20m$ room. Random goals are placed at least $10m$ away from the agents' initial positions. A crash event is registered

Figure 6.3: **The discriminator loss and reward curves.**

if laser-scan values fall below $0.5m$. During respawns, collision checks are made to ensure that each agent is not spawned in the vicinity of others.

Testing Setup. To evaluate the robustness of our policy in unseen crowd behaviors, we propose a novel set of testing conditions with the criteria that the movements of the other agents must be unique and not encountered during training. Also, we omit static obstacles as we focus on dynamic obstacle avoidance. In these environments, our agent interacts with other agents which exhibit dynamic movements beyond our control. For clarity, we refer to other agents as pedestrians for the remainder of this section. In total, we design six different pedestrian setups to be evaluated for each study:

1. *Non-homogeneous* (NH): To achieve non-homogeneous behaviors, each pedestrian utilizes a distinct policy instance, initialized with a unique seed, to generate varied experiences during training.

2. *Invisible* (IN): Similar to 1), but our robot is invisible to pedestrians achieved by lowering our agent's height below the pedestrians' sensors. This reflects the non-reactive individuals in the real world. We achieve this by lowering our agent's height below the pedestrians' sensor in the simulator.

3. *Variability* (VA): Similar to 2), pedestrians are invisible but receive random speed multipliers (0.5-1.5) at each episode start, representing real-world walking speed variability.

4. *Sub-optimal* (SO): Similar to 1), but the policy is trained for half the period (2.5k updates instead of 5k), simulating sub-optimal walking trajectories.

5. *Velocity-obstacle* (VO): We utilizes ground truth positions of all pedestrians to compute permissible velocities, following the method in [219].

6. *Social force* (SF): We use a force-based system to anticipate pedestrian movements following [220], and utilize ground positions for prediction similar to VO.

We consider a mixture of learning-based (NH,IN,VA,SO) and non-learning based (VO,SF) policies. Four of them (IN,SO,VO,SF) are challenging with the non-reactive pedestrians. All evaluations are repeated for 1000 episodes. If not stated explicitly, we set the number of robots $N = 5$ (1 agent and 4 pedestrians) and number of behaviors $M = 5$. Agents, pedestrians and goals are spawned similarly to training. During testing, each agent utilises a fixed behavior token, $h = 0$ for all episodes. Our primary metric is the 'success rate', without further classifying the non-successful episodes, as collisions are the primary reason instead of timeouts.

Metrics. We adopt the following metrics [17] to measure the performance and robustness of an agent.

- *Success rate*: we compute the success rate out of 1000 episodes. An episode is regarded as successful when the agent can reach within 0.5m of the desired goal in 50 seconds without any collision.

- *Extra distance*: this is computed as the difference between the agent's travel distance and the shortest distance between the start and goal positions. A smaller value indicates better efficiency for the agent.

- *Extra time*: this is computed as the difference between the agent's travel time and the shortest travel time. The shortest travel time is calculated as the shortest distance over the maximum linear velocity of the robot. A smaller value indicates better efficiency for the agent.

- *Average speed*: this is the average linear speed across all steps within an episode. A larger average speed implies better efficiency for the agent.

M	#Updates	#Updates/M	Pedestrian Type						Avg
			NH	IN	VA	SO	VO	SF	
1	5K	5K	0.63	0.55	0.59	0.52	0.59	0.43	0.55
5	5K	1K	**0.87**	**0.85**	**0.86**	**0.72**	**0.78**	**0.77**	**0.81**
10	5K	500	0.80	0.75	0.82	0.72	0.76	0.69	0.76
20	5K	250	0.61	0.63	0.54	0.59	0.59	0.51	0.58
10	10K	1K	0.88	0.86	0.86	0.76	0.77	0.78	0.82
20	20K	1K	0.89	0.87	0.85	0.77	0.78	0.79	0.82

Table 6.2: **Impact of the number of behaviors** M. Policies are evaluated under six diverse unseen pedestrian setups.

6.4.2 Training Stability

Figure 6.3 shows the stability of the discriminators and the intrinsic reward during training. The discriminator loss and intrinsic reward steadily improves until $\sim$ 700 updates, which then flattens until $\sim$ 4000 updates, possibly due to novel state-action exploration. Subsequently, both curves continue to improve again until the end of training where the main task has converged. The best advantage weight α, which combines the task and intrinsic reward is 0.1.

6.4.3 Impact of the Number of Behaviors

We assess diversity's impact on agent robustness and its scalability with behavior count M. When $M = 1$, all agents share the same behavior, equivalent to the baseline policy in [17]. Our results are presented in Table 6.2. We have the following three observations. First, having non-reactive pedestrians (NH,IN,VA,SO) is generally a more difficult task with fewer successful runs. Second, adding diversity ($M \neq 1$) outperforms the default policy ($M = 1$) for all pedestrian types, including the challenging non-reactive pedestrians. This validates the effectiveness of our proposed method. Third, the optimal number of behaviors is $M = 5$ and the effect of diversity starts to diminish as we scale to higher values of $M = 10$ and 20. We hypothesize the diminishing effect is resulted from less frequent sampling when M increases. To investigate this, we increase $num\_updates/M$ for $M = 10$ and 20 to match $M = 5$ and find that the performance of $M = 10$ and $M = 20$ could match that of $M = 5$, validating our hypothesis. However, this comes at an expense of more updates. Overall, $M = 5$ provides a good balance between creating good diversity and sample efficiency.

N	Diversity	Pedestrian Type						Avg
		NH	IN	VA	SO	VO	SF	
5	No	0.63	0.55	0.59	0.52	0.59	0.43	0.55
5	Yes	**0.87**	**0.85**	**0.86**	**0.72**	**0.78**	**0.77**	**0.81**
10	No	0.41	0.38	0.42	0.37	0.31	0.28	0.35
10	Yes	0.83	0.78	0.77	0.65	0.75	0.52	0.72
20	No	0.47	0.43	0.42	0.39	0.36	0.32	0.38
20	Yes	0.58	0.50	0.58	0.47	0.42	0.49	0.50

Table 6.3: **Impact of the number of agents N.**

Intrinsic Reward	α best	Pedestrian Type						Avg	ζ
		NH	IN	VA	SO	VO	SF		
$log[q_1(h \mid s,a)] - log[q_2(h \mid s)]$	0.10	**0.87**	**0.85**	**0.86**	**0.72**	**0.78**	**0.77**	**0.81**	1.10
$log[q(h \mid s,a)]$	0.01	0.88	0.77	0.86	0.70	0.75	0.72	0.78	0.48
$log[q(h \mid s)]$ [159]	0.03	0.75	0.75	0.69	0.71	0.73	0.58	0.70	0.17
None	0	0.63	0.55	0.59	0.52	0.59	0.43	0.55	0

Table 6.4: **Policies trained using different intrinsic rewards**. ζ is a measure of action diversity between agents

6.4.4 Scalability with number of agents

Here, we investigate the effect of increased numbers of agents N and report the results in Table 6.3. As the number of agents, N, increases to 10 and 20, the scenes become more crowded, making it harder for them to reach their goals. This negatively affects the convergence speed, as agents get restarted more frequently due to collisions or other obstacles. Despite this, by adding diversity to the policy, we achieve consistent performance improvements across all pedestrian cases.

6.4.5 Ablation Experiments: Intrinsic Rewards

In Section 6.3.1, we formulate a cost function to promote diversity among agents. In the formulation, we require that diverse actions performed for different h should arise for every state instead of exploiting only certain states. This is achieved using a regularizer as part of the intrinsic reward proposed in Eqn. (6.3). From ablation experiments reported in Table 6.4, we observe that the policy trained with the intrinsic reward containing the regularization term, $-log(q \mid s)$, outperforms the policy without this term. The performance improvement is consistent in all pedestrian setups including the challenging non-reactive pedestrians. Despite

Metrics	Policy	Pedestrian Type					
		NH	IN	VA	SO	VO	SF
Success Rate ↑	Basic	0.63	0.55	0.59	0.52	0.59	0.43
	Safe	0.86	0.77	0.81	0.67	0.69	0.61
	Ours	**0.87**	**0.85**	**0.86**	**0.72**	**0.78**	**0.77**
Extra Time (s) ↓	Basic	2.833 ± 2.439	3.366 ± 2.621	3.724 ± 2.219	2.511 ± 1.751	3.158 ± 2.121	2.997 ± 1.136
	Safe	5.041 ± 2.356	5.102 ± 2.719	5.248 ± 2.658	5.100 ± 2.335	5.217 ± 2.454	4.813 ± 2.348
	Ours	**2.712 ± 2.259**	**2.902 ± 2.671**	**2.714 ± 2.427**	**2.336 ± 0.995**	**2.119 ± 1.038**	**2.202 ± 1.344**
Extra Distance (m) ↓	Basic	4.811 ± 4.011	4.123 ± 5.637	5.717 ± 3.013	4.197 ± 5.873	5.887 ± 4.187	3.321 ± 3.899
	Safe	10.099 ± 4.452	10.734 ± 5.177	16.601 ± 5.235	10.116 ± 4.182	9.870 ± 3.946	10.024 ± 4.275
	Ours	**3.667 ± 3.587**	**3.930 ± 4.024**	**4.262 ± 4.489**	**3.217 ± 2.309**	**2.492 ± 1.719**	**2.662 ± 2.112**
Average Speed (m/s) ↑	Basic	0.919 ± 0.096	0.927 ± 0.088	0.917 ± 0.87	0.922 ± 0.068	0.920 ± 0.079	0.910 ± 0.087
	Safe	0.810 ± 0.091	0.795 ± 0.098	0.779 ± 0.100	0.811 ± 0.084	0.811 ± 0.084	0.786 ± 0.097
	Ours	**0.957 ± 0.059**	**0.955 ± 0.061**	**0.942 ± 0.076**	**0.960 ± 0.057**	**0.966 ± 0.039**	**0.965 ± 0.043**

Table 6.5: **Comparisons with baseline methods using different metrics averaged across 1000 episodes.**

this, the policy trained without regularization still outperforms the base policy without the intrinsic reward.

Next, we compare our method with state-space exploration based intrinsic rewards, DIYAN, from [159] which may implicitly encourage action diversity through novel state exploration. However, it still lacks action diversity compared to our proposed intrinsic reward in Eqn.(6.3), where the diversity of the action is explicitly encouraged. To measure the action diversity, we also introduce a new metric, ζ, using the KL divergence of action distributions between pairwise agents:

$$\zeta = \frac{1}{|\tau| N_{i \neq j}} \sum_{s \in \tau} \sum_{i \neq j} \mathrm{KL}\left(\pi(a|s, h=i) \| \pi(a|s, h=j)\right)$$

where τ denotes a trajectory. Specifically, we collect a trajectory of 1000 steps using the trained policy with no intrinsic reward. From Table 6.4, our proposed method achieves higher action diversity ζ than the state-space exploration based intrinsic rewards. Also, we observe that higher values of ζ get translated into higher robustness in unseen crowd behaviors, achieving a greater success rate.

6.4.6 Comparisons with Prior Work

We quantitatively compare our proposed behavior-conditioned policy with existing solutions to demonstrate its robustness. In particular, we set up the baseline method as described in [17], equivalent to our proposed method with $M = 1$. Additionally, we added a safe policy proposed in [23], which uses safety zone rewards to encourage safe behaviors, which could crash less

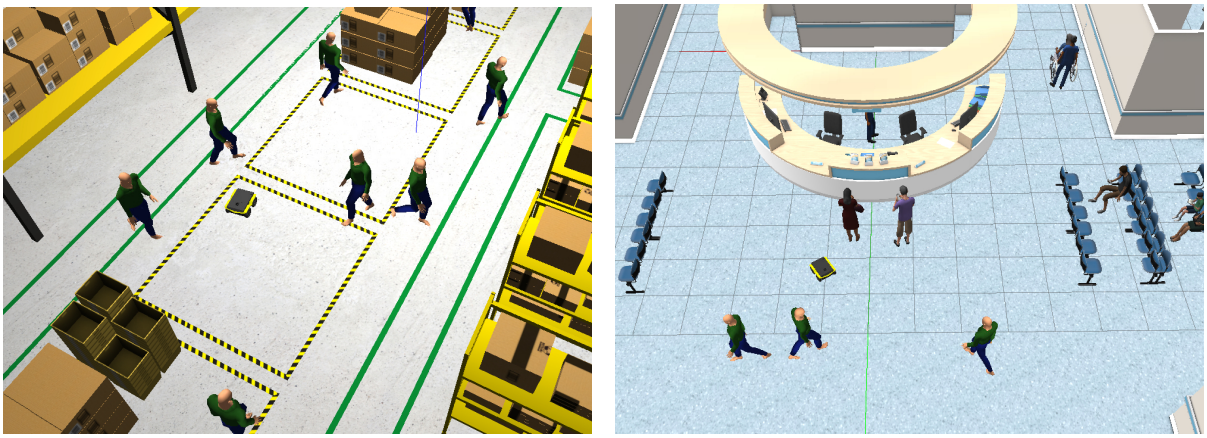


Figure 6.4: **Testing our method in Gazebo with more realistic scenarios**. Map settings: (Left) Warehouse (Right) Hospital

in unseen crowd movements. For our proposed method, we utilize the model trained with $M = 5$ and $N = 5$. Table 6.5 shows the comparison results against different metrics across 1000 episodes. Each metrics (success rate, extra time, extra distance, average speed) are similarly defined like in [17].

Our proposed method consistently outperforms others across various pedestrian types, suggesting robust strategies for handling diverse crowd behaviors effectively. While the safe policy achieves a higher success rate than the base policy, it slightly falls short of our proposed policy. Collisions primarily contribute to non-successful episodes, surpassing timeouts. The safe policy with a safety buffer performs well in reactive setups (NH, VA), closely matching our policy's results. However, it struggles in non-reactive setups (IN, SO, VO, SF). The conservative behavior of the safe policy reduces collisions but increases time and distance compared to our proposed policy, sacrificing other metrics. Specifically, both time and distance taken by the safe policy are more than double those of our proposed policy.

6.4.7 Qualitative Analysis

Next, we investigate if behavior-conditioned policies exhibit different behaviors to reach the desired goal. We record the behaviors of agents starting from the origin and reaching a fixed goal behind a static obstacle. The trajectories of the 5 agents can be seen in Figure 6.5. These agents exhibit different behaviors when conditioned on different tokens h. While passing the obstacle, some agents are left-inclined whereas some are right-inclined which is consistent with

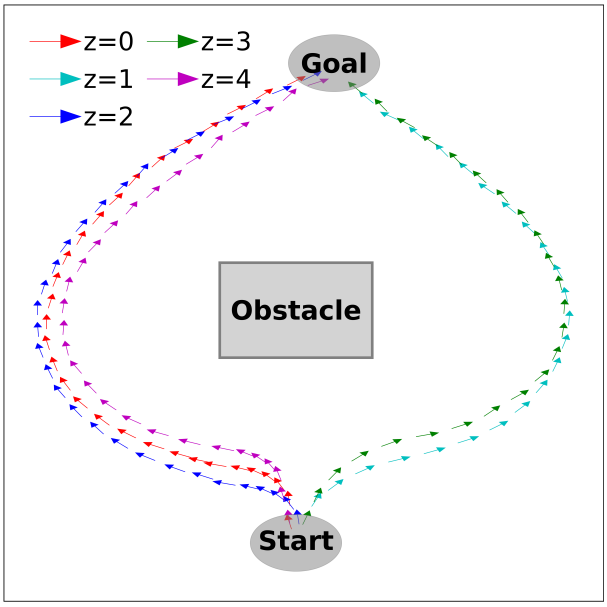


Figure 6.5: **Trajectories of diverse behaviors sampled from our behavior-conditioned policy**. Agents exhibit unique ways to reach the goal depending on sampled behaviors.

the motivation illustrated in Figure 6.1. Additionally, we also observe the velocity diversity which is represented by the length of the arrow. For example, the blue agent ($h = 2$) travels slightly faster than the purple agent ($h = 4$) when taking a slightly longer route. While the diversity may appear slight, deploying it with multiple agents generates vastly different training trajectories, enhancing agent robustness. More qualitative examples about the diversity among dynamic obstacles are available on https://youtu.be/EevMn2-ZNng.

6.4.8 Realistic Deployment

To validate our method in a more realistic setup, we deploy our best performing policy ($N = 5$, $M = 5$) on a Jackal robot in Gazebo simulator [64]. Our intent with these experiments is to illustrate the transferability of our framework across distinct simulation environments (from Stage to Gazebo) without policy modifications, emphasizing generalizability. We utilize two maps (warehouse and hospital) from Arena-ROSNAV-3D [221] as seen in Figure. 6.4. For each episode, we randomly assign the start and goal position with 3 to 8 pedestrians within an open area of each map ($\sim 10m \times 10m$). The pedestrians movements are simulated using the social force model [220].

Table 6.6 shows the success rate from 100 episodes with and without the diversity consideration. Our method proves to be equally effective for realistic scenes, outperforming the

Diversity	Warehouse	Hospital
No	0.40	0.37
Yes	**0.81**	**0.69**

Table 6.6: **Experimental results of Gazebo deployment.** Success rate out of 100 episodes

baseline method when diversity is used during training. More qualitative examples for realistic scenes are available on https://youtu.be/EevMn2-ZNng.

6.5 Conclusion

We introduce a framework to increase an agent's ability to generalize to unseen crowd behaviors by utilizing diverse behaviors in a sample-efficient manner. Adding diversity in a multi-agent framework implicitly provides each agent with a more varied range of experiences, hence increasing its generalizability of unseen crowd behaviors. We demonstrate the robustness of the proposed method in an extensive set of evaluation scenes containing challenging pedestrians' behaviors. We also validate the scalability of our solution and practicality in realistic scenes. Our experiments also demonstrate that our method improves the success rates without negatively affecting other important metrics.

Chapter 7

Conclusion

7.1 Summary

This thesis has provided a comprehensive investigation into methods for enhancing the practicality and effectiveness of RL-based local motion planners, focusing on bridging the reality gap and incorporating human-centered design principles. In the first part of the thesis, we addressed the dynamics mismatch challenge inherent in both online and offline RL settings. In the online RL setting, we presented a simple yet practical domain adaptation approach that compensates for dynamics mismatch through reward function adjustments with a novel dual action set. This method was proven to guarantee near-optimal policy performance in the target domain under mild assumptions. Experiments across various control tasks confirmed its effectiveness in leveraging source domain information to develop successful target domain policies, even with limited target domain transitions. Notably, a minimum of 10,000 offline samples from the target dataset was required to enable positive transfer, highlighting both the potential and limitations of this approach. Furthermore, we validated our hypothesis that uncertainty-based exploration is able to compensate state transitions with large variance. In the offline RL setting, we introduced a novel conditional diffusion model to solve the dynamics mismatch problem. The model incorporates a continuous dynamics score and an inverse-dynamics context to capture the underlying dynamics structure within the latent space. This enabled the model to learn effectively from both larger off-dynamics source datasets and limited, sub-optimal target datasets. Empirical results demonstrated that our method significantly outperformed existing baselines, and ablation studies confirmed the importance of each dynamics context in improving performance. The model also exhibited promising robustness in handling interpolation

scenarios, suggesting its potential for real-world applications with dynamic shifts in the target environment.

The second part of the thesis shifts its focus towards human-centered approaches, addressing the challenges of effective personalization and navigation in human-populated environments. First, we tackled the challenge of personalizing robot behaviors by developing a resource-efficient method that rapidly and accurately adapts to individual user preferences, eliminating the need for manually designed rewards. Specifically, our approach leverages a pretrain-finetune framework, allowing the network to learn a latent embedding that represents user preferences. This pretraining process enables rapid adaptation by updating only the embedding while keeping the rest of the network parameters fixed. Evaluation results demonstrated superior adaptation accuracy with less data compared to other methods, proving its effectiveness in both simulated and human-annotated benchmarks. In addition to attaining high accuracy, our proposed method exhibited greater stability across update steps compared to RLHF and PPO, further highlighting its potential for real-world deployment. Lastly, we addressed the challenge of developing robust and adaptable agents capable of navigating unpredictable environments with unforeseen crowd movements. We proposed a method to foster diversity within multi-agent learning systems. By introducing behavior-conditioned policies with distinct behavior tokens, we successfully encouraged diverse behaviors among agents, a result we verified qualitatively. This increased diversity exposed each agent to a broader range of experiences, which enhances their ability to generalize to novel crowd behaviors. Experiments showed a significant improvement in success rates without negatively impacting other essential metrics, even in high-density environments and realistic Gazebo simulations.

In conclusion, this thesis contributes to the advancement of RL-based local motion planning, with potential implications for the development of intelligent robotic systems that can better integrate with human society. We believe that our approaches to tackling dynamics mismatch have shown promising results and could significantly improve the real-world applicability of these methods. The work presented here seeks to address the challenges of navigation in unpredictable environments and adaptation to user preferences, with the goal of creating safer and more efficient robots. While this research represents a step forward in the field, there remains much to be explored in the ongoing quest to fully realize the potential of RL in shaping the future of robotics and human-robot interaction.

7.2 Broader Impact

Here, we discuss the broader impact of our research beyond mobile robots:

Reducing the dynamics mismatch between source and target domains can be extended to general robotics and even non-robotic domains. In general robotics, our approaches can benefit several key areas. Industrial robots can leverage these methods when transferring policies from simulation to real-world tasks involving materials with varying physical properties, such as cable manipulation, fabric handling, and precision assembly. Medical robots could use these techniques to bridge the crucial gap between simulated and real surgical procedures, where tissue properties and tool interactions vary significantly. For soft robotics, our methods address the challenging dynamics of deformable materials, where simplified simulation models often fail to capture the highly nonlinear behaviors of real-world applications. Beyond robotics, our sim-to-real principles have valuable applications in autonomous vehicles and decision-making systems. In autonomous driving, simulation-trained models often face performance issues due to mismatches in vehicle physics and road conditions, but our approaches can improve real-world reliability. Similarly, in industrial automation, control systems initially designed and tested in simulation can benefit from our methods, enhancing their transition to operational environments and resulting in more robust and efficient automation processes.

Personalization in the context of reinforcement learning and sequential decision-making also has the potential for significant impact beyond mobile robots and traditional robotics applications. Assistive robots, such as eldercare or rehabilitation devices, can use preference learning to align their actions with individual needs and comfort, improving quality of life. Similarly, assistive technologies, like prosthetics or exoskeletons, can be personalized to match user preferences for movement and comfort, enhancing usability and effectiveness. Beyond robotics, preference learning has transformative potential in non-robotic domains such as education and healthcare. For example, adaptive learning platforms can personalize educational content, pacing, and feedback to suit diverse learning styles. In healthcare, preference-aware RL can align medical decision-making with clinician priorities and patient-centric care. On a broader societal level, preference learning can influence policy design and governance by balancing competing preferences among diverse groups, leading to more inclusive and equitable decision-making processes.

Generalizing and learning diverse behaviors for crowd movement can significantly enhance swarm robotics for drones and autonomous vehicles. In drone swarms, understanding and replicating behaviors like flocking, dispersion, and collective decision-making enables more efficient and adaptive operations. For instance, in delivery systems, swarm drones can utilize crowd-inspired behaviors to optimize route selection, avoid mid-air collisions, and adapt to changing traffic in urban airspaces, ensuring faster and safer deliveries. Similarly, in autonomous vehicles, crowd movement principles can improve navigation and coordination in complex urban settings. By learning diverse behaviors—such as merging into traffic, yielding to pedestrians, or forming vehicle platoons—self-driving cars can operate more safely and efficiently, especially in mixed-traffic scenarios involving vehicles, cyclists, and pedestrians.

7.3 Future Work

Our research has established theoretical foundations for improving RL-based local motion planners, addressing the reality gap and integrating human-centered design principles. However, several aspects of our work require further investigation and refinement:

In addressing dynamics mismatch challenges in online and offline setting, we propose a few ideas. In the **online** settings, further research is needed to explore performance based on the optimality of the behavioral policy. Our current approach uses the source policy as the behavioral policy, but investigation into less optimal policies is necessary. Additionally, we need to analyze the relationship between the effectiveness of reward adjustment and the extent of the dynamics gap between source and target environments, as well as understand the limitations of this approach as the gap widens. Next, the uncertainty-based exploration technique shows promise, particularly in handling states with epistemic errors in the DARC classifiers, and could be extended to more advanced formulations or other applications. In the **offline** settings, our current setup is limited to single source off-dynamics datasets. Future studies should explore the use of multiple off-dynamics datasets, potentially incorporating a multi-class classifier formulation for domain classifiers, to enhance model generalization across diverse environments. This approach could be viewed as a form of meta-learning, where the model learns to adapt to various dynamics distributions. Lastly, while the proposed dynamics score and inverse dynamics context proved effective, utilizing other forms of dynamics contexts with the conditional DPMs remains under-explored.

Next, we propose several ideas for human-centered applications. For the **personalization** topic, our newly designed experiment based on Quality Diversity prioritizes diversity over optimality, better reflecting real-world scenarios for preference learning. While this represents a significant step forward, the current study is limited in the variety of tasks and number of users involved. To address these limitations, we propose expanding our research to incorporate larger human-annotated datasets encompassing a broader range of tasks. This expansion could provide a more comprehensive understanding of human preferences across a wider range of scenarios, enhancing the robustness and applicability of our preference learning models. Additionally, we introduced a novel pretraining approach for the diffusion model, utilizing masked trajectories and spectral normalization without relying on explicit supervision. While promising, this method leaves room for further exploration in pretraining design. Future research could investigate alternative pretraining strategies, architectures, or regularization techniques to potentially enhance this crucial step. In the context of generalization of **human pedestrians** behaviors, our current approach calculates intrinsic rewards based on single-step predictions. Future research could explore extending this to longer time horizons, potentially capturing more complex behavioral patterns and improving long-term planning. Additionally, we propose integrating our diversity concept with imitation learning using real pedestrian datasets. While imitation learning closely mimics human movement patterns, it often struggles with overfitting and poor generalization. By incorporating diversity into the imitation learning process, we could potentially leverage the strengths of both approaches, creating agents that are more robust and adaptable in human-populated environments.

Finally, while our methods have been empirically validated against numerous strong baselines in Mujoco and Gazebo simulations, these environments do not fully replicate real-world complexities. Future research should prioritize testing robots on actual hardware in real-world settings to validate and refine our approaches, a crucial step in understanding the true capabilities and limitations of our methods.

Bibliography

[1] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[2] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[3] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: a survey," *Autonomous Robots*, vol. 46, no. 5, pp. 569–597, 2022.

[4] H.-y. Zhang, W.-m. Lin, and A.-x. Chen, "Path planning for the mobile robot: A review," *Symmetry*, vol. 10, no. 10, p. 450, 2018.

[5] M. Ben-Ari, F. Mondada, M. Ben-Ari, and F. Mondada, "Robots and their applications," *Elements of robotics*, pp. 1–20, 2018.

[6] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[7] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *2015 european control conference (ECC)*. IEEE, 2015, pp. 3352–3357.

[8] N. A. K. Zghair and A. S. Al-Araji, "A one decade survey of autonomous mobile robot systems," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 6, p. 4891, 2021.

[9] J. A. Abdulsaheb and D. J. Kadhim, "Classical and heuristic approaches for mobile robot path planning: A survey," *Robotics*, vol. 12, no. 4, p. 93, 2023.

[10] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[11] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[12] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 797–803.

[13] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[15] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 285–292.

[16] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *International Conference on Intelligent Robots and Systems*, 2018.

[17] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6252–6259.

[18] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6015–6022.

[19] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[20] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, "Curiosity-driven exploration for mapless navigation with deep reinforcement learning," *arXiv preprint arXiv:1804.00456*, 2018.

[21] T. Fan, P. Long, W. Liu, J. Pan, R. Yang, and D. Manocha, "Learning resilient behaviors for navigation under uncertainty," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5299–5305.

[22] Q. Tan, T. Fan, J. Pan, and D. Manocha, "Deepmnavigate: Deep reinforced multi-robot navigation unifying local & global collision avoidance," in *International Conference on Intelligent Robots and Systems*, 2020.

[23] J. Jin, N. M. Nguyen, N. Sakib, D. Graves, H. Yao, and M. Jagersand, "Mapless navigation among dynamics with social-safety-awareness: a reinforcement learning approach from 2d laser scans," in *IEEE international conference on robotics and automation*, 2020.

[24] Y. Cui, H. Zhang, Y. Wang, and R. Xiong, "Learning world transition model for socially aware robot navigation," in *IEEE International Conference on Robotics and Automation*, 2021.

[25] Z. Xu, X. Xiao, G. Warnell, A. Nair, and P. Stone, "Machine learning methods for local motion planning: A study of end-to-end vs. parameter learning," in *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2021, pp. 217–222.

[26] M. Sun, F. Baldini, P. Trautman, and T. Murphey, "Move beyond trajectories: Distribution space coupling for crowd navigation," *arXiv preprint arXiv:2106.13667*, 2021.

[27] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *ICRA*, 2018.

[28] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv preprint*, 2017.

[29] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *ICRA*, 2019.

[30] M. K. Lee, J. Forlizzi, S. Kiesler, P. Rybski, J. Antanitis, and S. Savetsila, "Personalization in hri: A longitudinal field experiment," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, 2012, pp. 319–326.

[31] M.-H. Huang and R. T. Rust, "Engaged to a robot? the role of ai in service," *Journal of Service Research*, vol. 24, no. 1, pp. 30–41, 2021.

[32] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[33] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv preprint arXiv:1511.03791*, 2015.

[34] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[35] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[36] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *2010 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2010, pp. 3232–3237.

[37] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[38] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, 2015.

[40] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[41] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[42] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[44] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[45] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[46] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint*, 2015.

[47] M. Zare, P. M. Kebria, and A. Khosravi, "Leveraging optimal transport for enhanced offline reinforcement learning in surgical robotic environments," *arXiv preprint arXiv:2310.08841*, 2023.

[48] K. Fan, Z. Chen, G. Ferrigno, and E. De Momi, "Learn from safe experience: Safe reinforcement learning for task automation of surgical robot," *IEEE Transactions on Artificial Intelligence*, 2024.

[49] R.-J. Qin, X. Zhang, S. Gao, X.-H. Chen, Z. Li, W. Zhang, and Y. Yu, "Neorl: A near real-world benchmark for offline reinforcement learning," *NeurIPS*, 2022.

[50] N. Lee and J. Moon, "Offline reinforcement learning for automated stock trading," *IEEE Access*, 2023.

[51] H.-H. Tseng, Y. Luo, S. Cui, J.-T. Chien, R. K. Ten Haken, and I. E. Naqa, "Deep reinforcement learning for automated radiation adaptation in lung cancer," *Medical physics*, 2017.

[52] X. Nie, E. Brunskill, and S. Wager, "Learning when-to-treat policies," *Journal of the American Statistical Association*, 2021.

[53] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *ICRA*, 2019.

[54] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *IEEE international conference on robotics and automation*, 2018.

[55] L. Sun, C. Peng, W. Zhan, and M. Tomizuka, "A fast integrated planning and control framework for autonomous driving via imitation learning," in *Dynamic Systems and Control Conference*, 2018.

[56] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," *NeurIPS*, 2019.

[57] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *ICML*, 2019.

[58] N. Y. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller, "Keep doing what worked: Behavioral modelling priors for offline reinforcement learning," *arXiv preprint arXiv:2002.08396*, 2020.

[59] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[60] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *NeurIPS*, 2019.

[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[62] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2371–2378.

[63] Y. Zhou, S. Li, and J. Garcke, "R-sarl: Crowd-aware navigation based deep reinforcement learning for nonholonomic robot in complex environments," *arXiv preprint arXiv:2105.13409*, 2021.

[64] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[65] J. v. d. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[66] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[67] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[68] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=PxTIG12RRHS

[69] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[70] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36 479–36 494, 2022.

[71] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[72] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741*, 2021.

[73] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," *arXiv preprint arXiv:2009.09761*, 2020.

[74] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov, "Grad-tts: A diffusion probabilistic model for text-to-speech," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8599–8608.

[75] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," *arXiv:2204.03458*, 2022.

[76] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017.

[77] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *NeurIPS*, 2017.

[78] J. Ho, "Classifier-free diffusion guidance," *ArXiv*, vol. abs/2207.12598, 2022.

[79] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Cascaded diffusion models for high fidelity image generation," *JMLR*, 2022.

[80] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool, "Repaint: Inpainting using denoising diffusion probabilistic models," in *CVPR*, 2022.

[81] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *CVPR*, 2023.

[82] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang, "Geodiff: A geometric diffusion model for molecular conformation generation," in *ICLR*, 2021.

[83] Z. Lyu, Z. Kong, X. Xudong, L. Pan, and D. Lin, "A conditional point diffusion-refinement paradigm for 3d point cloud completion," in *ICLR*, 2021.

[84] 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 *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=sP1fo2K9DFG

[85] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021.

[86] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *International Conference on Machine Learning*, 2022.

[87] J. Hu, Y. Sun, S. Huang, S. Guo, H. Chen, L. Shen, L. Sun, Y. Chang, and D. Tao, "Instructed diffuser with temporal condition guidance for offline reinforcement learning," *arXiv preprint arXiv:2306.04875*, 2023.

[88] Z. Zhu, M. Liu, L. Mao, B. Kang, M. Xu, Y. Yu, S. Ermon, and W. Zhang, "Madiff: Offline multi-agent learning with diffusion models," *arXiv preprint arXiv:2305.17330*, 2023.

[89] F. Ni, J. Hao, Y. Mu, Y. Yuan, Y. Zheng, B. Wang, and Z. Liang, "Metadiffuser: Diffusion model as conditional planner for offline meta-rl," *arXiv preprint arXiv:2305.19923*, 2023.

[90] H. He, C. Bai, K. Xu, Z. Yang, W. Zhang, D. Wang, B. Zhao, and X. Li, "Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning," *arXiv preprint arXiv:2305.18459*, 2023.

[91] Y. Zheng, J. Li, D. Yu, Y. Yang, S. E. Li, X. Zhan, and J. Liu, "Safe offline reinforcement learning with feasibility-guided diffusion model," *arXiv preprint arXiv:2401.10700*, 2024.

[92] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[93] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *ICML*, 2021.

[94] W. Xiao, T.-H. Wang, C. Gan, and D. Rus, "Safediffuser: Safe planning with diffusion probabilistic models," *arXiv preprint arXiv:2306.00148*, 2023.

[95] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[96] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[97] Y. Lee, J. Yang, and J. J. Lim, "Learning to coordinate manipulation skills via skill behavior diversification," in *International conference on learning representations*, 2020.

[98] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *ICRA*, 2017.

[99] S. Liu, K. C. See, K. Y. Ngiam, L. A. Celi, X. Sun, and M. Feng, "Reinforcement learning for clinical decision support in critical care: comprehensive review," *Journal of medical Internet research*, 2020.

[100] A. Charpentier, R. Elie, and C. Remlinger, "Reinforcement learning in economics and finance," *Computational Economics*, 2021.

[101] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2019.

[102] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[103] T. Du, J. Hughes, S. Wah, W. Matusik, and D. Rus, "Underwater soft robot modeling and control with differentiable simulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4994–5001, 2021.

[104] J. Hanna and P. Stone, "Grounded action transformation for robot learning in simulation," in *AAAI*, 2017.

[105] H. Karnan, S. Desai, J. P. Hanna, G. Warnell, and P. Stone, "Reinforced grounded action transformation for sim-to-real transfer," in *IROS*, 2020.

[106] P. F. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," *arXiv preprint*, 2016.

[107] B. Eysenbach, S. Chaudhari, S. Asawa, S. Levine, and R. Salakhutdinov, "Off-dynamics reinforcement learning: Training for transfer with domain classifiers," in *ICLR*, 2020.

[108] H. Niu, Y. Qiu, M. Li, G. Zhou, J. HU, X. Zhan *et al.*, "When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning," *NeurIPS*, 2022.

[109] Y. Jiang, T. Zhang, D. Ho, Y. Bai, C. K. Liu, S. Levine, and J. Tan, "Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning," in *ICRA*, 2021.

[110] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Conference on Robot Learning*. PMLR, 2020, pp. 1162–1176.

[111] M. Mozian, J. C. G. Higuera, D. Meger, and G. Dudek, "Learning domain randomization distributions for training robust locomotion policies," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6112–6117.

[112] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[113] B. Eysenbach, A. Khazatsky, S. Levine, and R. R. Salakhutdinov, "Mismatched no more: Joint model-policy optimization for model-based rl," *NeurIPS*, 2022.

[114] Y. Kang, J. Liu, X. Cao, and D. Wang, "Off-dynamics inverse reinforcement learning from hetero-domain," *arXiv preprint*, 2021.

[115] J. Liu, H. Shen, D. Wang, Y. Kang, and Q. Tian, "Unsupervised domain adaptation with dynamics-aware rewards in reinforcement learning," *NeurIPS*, 2021.

[116] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *ICML*, 2015.

[117] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *NeurIPS*, 2018.

[118] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," in *ICLR*, 2018.

[119] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," *NeurIPS*, 2016.

[120] K. Ghasemipour, S. S. Gu, and O. Nachum, "Why so pessimistic? estimating uncertainties for offline rl through ensembles, and why their independence matters," *NeurIPS*, 2022.

[121] G. An, S. Moon, J.-H. Kim, and H. O. Song, "Uncertainty-based offline reinforcement learning with diversified q-ensemble," *NeurIPS*, 2021.

[122] X. Liang, K. Shu, K. Lee, and P. Abbeel, "Reward uncertainty for exploration in preference-based reinforcement learning," in *Deep RL Workshop NeurIPS 2021*, 2021.

[123] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *NeurIPS*, 2017.

[124] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IROS*, 2012.

[125] S. Desai, I. Durugkar, H. Karnan, G. Warnell, J. Hanna, and P. Stone, "An imitation from observation approach to transfer learning with dynamics mismatch," *NeurIPS*, 2020.

[126] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," in *Reinforcement learning: State-of-the-art*, 2012.

[127] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *CoLR*, 2022.

[128] J. Liu, Z. Hongyin, and D. Wang, "Dara: Dynamics-aware reward augmentation in offline reinforcement learning," in *ICLR*, 2021.

[129] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, 2009.

[130] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *ICANN*, 2018.

[131] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *JMLR*, 2020.

[132] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, "Transfer learning in natural language processing," in *NAAC:Tutorials*, 2019.

[133] K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal, "Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection," *Construction and building materials*, 2017.

[134] P. F. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model. corr (2016)," 2016.

[135] J. Ho and S. Ermon, "Generative adversarial imitation learning," *NeurIPS*, 2016.

[136] Z. Liang, Y. Mu, M. Ding, F. Ni, M. Tomizuka, and P. Luo, "Adaptdiffuser: Diffusion models as adaptive self-evolving planners," in *International Conference on Machine Learning*, 2023.

[137] I. Bae, J. Moon, J. Jhung, H. Suk, T. Kim, H. Park, J. Cha, J. Kim, D. Kim, and S. Kim, "Self-driving like a human driver instead of a robocar: Personalized comfortable driving experience for autonomous vehicles," *arXiv preprint arXiv:2001.03908*, 2020.

[138] X. He and C. Lv, "Toward personalized decision making for autonomous vehicles: a constrained multi-objective reinforcement learning technique," *Transportation research part C: emerging technologies*, vol. 156, p. 104352, 2023.

[139] E. OhnBar, K. Kitani, and C. Asakawa, "Personalized dynamics models for adaptive assistive navigation systems," in *Conference on Robot Learning*. PMLR, 2018, pp. 16–39.

[140] Y. Gao, W. Barendregt, M. Obaid, and G. Castellano, "When robot personalisation does not help: Insights from a robot-supported learning study," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2018, pp. 705–712.

[141] C. Moro, G. Nejat, and A. Mihailidis, "Learning and personalizing socially assistive robot behaviors to aid with activities of daily living," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 7, no. 2, pp. 1–25, 2018.

[142] Y. Wen, J. Si, A. Brandt, X. Gao, and H. H. Huang, "Online reinforcement learning control for the personalization of a robotic knee prosthesis," *IEEE transactions on cybernetics*, vol. 50, no. 6, pp. 2346–2356, 2019.

[143] B. Woodworth, F. Ferrari, T. E. Zosa, and L. D. Riek, "Preference learning in assistive robotics: Observational repeated inverse reinforcement learning," in *Machine learning for healthcare conference*. PMLR, 2018, pp. 420–439.

[144] X. Tu, M. Li, M. Liu, J. Si, and H. H. Huang, "A data-driven reinforcement learning solution framework for optimal and adaptive personalization of a hip exoskeleton," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 610–10 616.

[145] Q. Zhang, V. Nalam, X. Tu, M. Li, J. Si, M. D. Lewek, and H. H. Huang, "Imposing healthy hip motion pattern and range by exoskeleton control for individualized assistance," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 126–11 133, 2022.

[146] V. Nalam, X. Tu, M. Li, J. Si, and H. H. Huang, "Admittance control based human-in-the-loop optimization for hip exoskeleton reduces human exertion during walking," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6743–6749.

[147] C. Wirth, R. Akrour, G. Neumann, J. Fürnkranz *et al.*, "A survey of preference-based reinforcement learning methods," *Journal of Machine Learning Research*, vol. 18, no. 136, pp. 1–46, 2017.

[148] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in neural information processing systems*, vol. 30, 2017.

[149] A. Wilson, A. Fern, and P. Tadepalli, "A bayesian approach for policy learning from trajectory preference queries," *Advances in neural information processing systems*, vol. 25, 2012.

[150] A. Haydari and Y. Yılmaz, "Deep reinforcement learning for intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 11–32, 2020.

[151] C. Yu, J. Liu, S. Nemati, and G. Yin, "Reinforcement learning in healthcare: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–36, 2021.

[152] Z. Zhang, D. Zhang, and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," *CSEE Journal of Power and Energy Systems*, vol. 6, no. 1, pp. 213–225, 2019.

[153] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Computers & Chemical Engineering*, vol. 139, p. 106886, 2020.

[154] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[155] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[156] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-maron, M. Giménez, Y. Sulsky, J. Kay, J. T. Springenberg *et al.*, "A generalist agent," *Transactions on Machine Learning Research*, 2022.

[157] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2020.

[158] K. Lee, L. M. Smith, and P. Abbeel, "Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6152–6163.

[159] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," in *International Conference on Learning Representations*, 2019.

[160] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=68n2s9ZJWF8

[161] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 20 132–20 145, 2021.

[162] W. B. Knox, S. Hatgis-Kessell, S. Booth, S. Niekum, P. Stone, and A. Allievi, "Models of human preference for learning reward functions," *arXiv preprint arXiv:2206.02231*, 2022.

[163] D. J. Hejna III and D. Sadigh, "Few-shot preference learning for human-in-the-loop rl," in *Conference on Robot Learning*. PMLR, 2023, pp. 2014–2025.

[164] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in atari," *Advances in neural information processing systems*, vol. 31, 2018.

[165] C. Kim, J. Park, J. Shin, H. Lee, P. Abbeel, and K. Lee, "Preference transformer: Modeling human preferences using transformers for RL," in *International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=Peot1SFDX0

[166] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," *arXiv preprint arXiv:2305.18290*, 2023.

[167] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[168] W. Xia, Y. Zhang, Y. Yang, J.-H. Xue, B. Zhou, and M.-H. Yang, "Gan inversion: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 3, pp. 3121–3138, 2022.

[169] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*. Springer, 2016, pp. 597–613.

[170] R. Abdal, Y. Qin, and P. Wonka, "Image2stylegan: How to embed images into the stylegan latent space?" in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 4432–4441.

[171] ——, "Image2stylegan++: How to edit the embedded images?" in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8296–8305.

[172] J. Gu, Y. Shen, and B. Zhou, "Image processing using multi-code gan prior," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3012–3021.

[173] E. Richardson, Y. Alaluf, O. Patashnik, Y. Nitzan, Y. Azar, S. Shapiro, and D. Cohen-Or, "Encoding in style: a stylegan encoder for image-to-image translation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2287–2296.

[174] J. Zhu, Y. Shen, D. Zhao, and B. Zhou, "In-domain gan inversion for real image editing," in *European conference on computer vision*. Springer, 2020, pp. 592–608.

[175] O. Tov, Y. Alaluf, Y. Nitzan, O. Patashnik, and D. Cohen-Or, "Designing an encoder for stylegan image manipulation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–14, 2021.

[176] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[177] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or, "An image is worth one word: Personalizing text-to-image generation using textual inversion," *arXiv preprint arXiv:2208.01618*, 2022.

[178] J. Kreutzer, J. Uyheng, and S. Riezler, "Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning," *arXiv preprint arXiv:1805.10627*, 2018.

[179] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano, "Learning to summarize with human feedback," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3008–3021, 2020.

[180] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving, "Fine-tuning language models from human preferences," *arXiv preprint arXiv:1909.08593*, 2019.

[181] R. Ramamurthy, P. Ammanabrolu, K. Brantley, J. Hessel, R. Sifa, C. Bauckhage, H. Hajishirzi, and Y. Choi, "Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization," *arXiv preprint arXiv:2210.01241*, 2022.

[182] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[183] M. G. Azar, M. Rowland, B. Piot, D. Guo, D. Calandriello, M. Valko, and R. Munos, "A general theoretical paradigm to understand learning from human preferences," *arXiv preprint arXiv:2310.12036*, 2023.

[184] K. Ethayarajh, W. Xu, N. Muennighoff, D. Jurafsky, and D. Kiela, "Kto: Model alignment as prospect theoretic optimization," *arXiv preprint arXiv:2402.01306*, 2024.

[185] H. Xu, A. Sharaf, Y. Chen, W. Tan, L. Shen, B. Van Durme, K. Murray, and Y. J. Kim, "Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation," *arXiv preprint arXiv:2401.08417*, 2024.

[186] Y. Zhao, R. Joshi, T. Liu, M. Khalman, M. Saleh, and P. J. Liu, "Slic-hf: Sequence likelihood calibration with human feedback," *arXiv preprint arXiv:2305.10425*, 2023.

[187] G. An, J. Lee, X. Zuo, N. Kosaka, K.-M. Kim, and H. O. Song, "Direct preference-based policy optimization without reward modeling," *Advances in Neural Information Processing Systems*, vol. 36, pp. 70 247–70 266, 2023.

[188] B. Wallace, M. Dang, R. Rafailov, L. Zhou, A. Lou, S. Purushwalkam, S. Ermon, C. Xiong, S. Joty, and N. Naik, "Diffusion model alignment using direct preference optimization," 2023.

[189] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International conference on machine learning*. PMLR, 2019, pp. 2790–2799.

[190] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[191] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *International Conference on Machine Learning*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248965046

[192] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, p. 202845, 2016.

[193] E. Conti, V. Madhavan, F. Petroski Such, J. Lehman, K. Stanley, and J. Clune, "Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents," *Advances in neural information processing systems*, vol. 31, 2018.

[194] S. Wu, J. Yao, H. Fu, Y. Tian, C. Qian, Y. Yang, Q. Fu, and Y. Wei, "Quality-similar diversity via population based reinforcement learning," in *The Eleventh International Conference on Learning Representations*, 2022.

[195] C. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld, and J. Oh, "Core challenges of social robot navigation: A survey," *ACM Transactions on Human-Robot Interaction*, vol. 12, no. 3, pp. 1–39, 2023.

[196] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018.

[197] M. Caruso, E. Regolin, F. J. Camerota Verdù, S. A. Russo, L. Bortolussi, and S. Seriani, "Robot navigation in crowded environments: A reinforcement learning approach," *Machines*, vol. 11, no. 2, p. 268, 2023.

[198] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5671–5677.

[199] S. S. Samsani, H. Mutahira, and M. S. Muhammad, "Memory-based crowd-aware robot navigation using deep reinforcement learning," *Complex & Intelligent Systems*, vol. 9, no. 2, pp. 2147–2158, 2023.

[200] D. Martinez-Baselga, L. Riazuelo, and L. Montano, "Improving robot navigation in crowded environments using intrinsic rewards," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9428–9434.

[201] T. Fan, X. Cheng, J. Pan, P. Long, W. Liu, R. Yang, and D. Manocha, "Getting robots unfrozen and unlost in dense pedestrian crowds," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1178–1185, 2019.

[202] J. Liang, U. Patel, A. J. Sathyamoorthy, and D. Manocha, "Crowd-steer: Realtime smooth and collision-free robot navigation in densely crowded scenarios trained using high-fidelity simulation," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 4221–4228.

[203] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, "Where to go next: Learning a subgoal recommendation policy for navigation in dynamic environments," *IEEE Robotics and Automation Letters*, 2021.

[204] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The international journal of robotics research*, vol. 17, no. 7, pp. 760–772, 1998.

[205] Y. Wang, H. He, and C. Sun, "Learning to navigate through complex dynamic environment with modular deep reinforcement learning," *IEEE Transactions on Games*, vol. 10, no. 4, pp. 400–412, 2018.

[206] S. H. Semnani, H. Liu, M. Everett, A. De Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3221–3226, 2020.

[207] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *The International Journal of Robotics Research*, 2020.

[208] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International conference on machine learning*. PMLR, 2017, pp. 1352–1361.

[209] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *NeurIPS*, vol. 30, 2017.

[210] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[211] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 7234–7284, 2020.

[212] K. R. McKee, I. Gemp, B. McWilliams, E. A. Duèñez-Guzmán, E. Hughes, and J. Z. Leibo, "Social diversity and social preferences in mixed-motive reinforcement learning," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 869–877.

[213] C. Li, T. Wang, C. Wu, Q. Zhao, J. Yang, and C. Zhang, "Celebrating diversity in shared multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 3991–4002, 2021.

[214] D. Barber and F. Agakov, "The im algorithm: a variational approach to information maximization," *NeurIPS*, 2004.

[215] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[216] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[217] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm intelligence*, vol. 2, pp. 189–208, 2008.

[218] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[219] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium ISRR*. Springer, 2011, pp. 3–19.

[220] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[221] L. Kästner, T. Bhuiyan, T. A. Le, E. Treis, J. Cox, B. Meinardus, J. Kmiecik, R. Carstens, D. Pichel, B. Fatloun, N. Khorsandi, and J. Lambrecht, "Arena-bench: A benchmarking suite for obstacle avoidance approaches in highly dynamic environments," 2022. [Online]. Available: https://arxiv.org/abs/2206.05728