TOWARDS REAL-WORLD SOCIAL ROBOT NAVIGATION WITH DEEP REINFORCEMENT LEARNING AND IMITATION LEARNING

by

James Ronald Han

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
University of Toronto Institute for Aerospace Studies
University of Toronto

Abstract

Towards Real-world Social Robot Navigation with Deep Reinforcement Learning and Imitation Learning

James Ronald Han
Master of Applied Science
University of Toronto Institute for Aerospace Studies
University of Toronto
2025

Deep Reinforcement Learning (DRL) and Imitation Learning (IL) are promising approaches for learning socially compliant robot-navigation behaviors. These methods enable policies to be learned directly from interaction, avoiding the need to explicitly model human behavior. We propose a decision process formulation that unifies path tracking and human avoidance within a corridor abstraction. We develop a real-time, single-sensor system with robust perception-to-control integration for reliable deployment. We then introduce Deep Residual Model Predictive Control (DR-MPC), a sample-efficient DRL method that leverages MPC-based path tracking. DR-MPC significantly outperforms classical and residual DRL baselines in simulation, and we further validate its feasibility by training it directly in the real world. Lastly, we identify key architectural and training decisions that substantially improve Behaviour Cloning, achieving a three-times improvement in success rate. While these results are promising, challenges remain in reducing real-world data requirements and capturing the multi-modal nature of social navigation.

Acknowledgements

There are so many people who made this journey possible, so there’s no wrong place to start. I would like to thank my supervisors. Professor Tim Barfoot, thank you for your invaluable guidance and for consistently pushing me to become a better researcher. Although we occasionally had different perspectives on the role of learning and RL, our discussions deeply shaped my understanding of its place in robotics—and more broadly, the importance of grounding learning methods in the realities of the real world and other approaches. Professor Nick Rhinehart, your insights into ML and RL were instrumental not only in shaping our research but in deepening my understanding and appreciation of the field.

I’m also grateful to Hugues Thomas for the many thoughtful discussions that contributed to the development of the DR-MPC algorithm. Thanks to Alec Krawciw for being a fantastic mentor for real-world robots, and to Trevor Ablett for helping me expand my RL knowledge.

A heartfelt thank you to my friends who supported the real-world experiments with DR-MPC. Ian, thank you for being a wonderful roommate and an even better friend. Your support through both the highs and lows of this Master’s journey meant the world to me.

I am also grateful to my colleagues in the RI, my lab, and other labs whose support has made this journey more enjoyable.

Finally, to my family: thank you for shaping me into the person I am today. The values you instilled in me—my mindset toward learning, hard work, consistency, and life—have been invaluable to the completion of this thesis.

Contents

List of Tables

List of Figures

List of Acronyms

ATR	Actuation Termination Rate
BC	Behaviour Cloning
BEV	Bird’s Eye View
CCW	Counter-clockwise
CE	Cross Entropy
CNN	Convolutional Neural Network
CVMM	Constant Velocity Motion Model
CW	Clockwise
DDS	Data Distribution Service
DRL	Deep Reinforcement Learning
DR-MPC	Deep Residual Model Predictive Control
DSAC	Discrete Soft Actor-Critic
DWA	Dynamic Window Approach
EDR	End Deviation Rate
FOV	Field-of-view
GNN	Graph Neural Network
HA	Human Avoidance
HAN	Human Avoidance Network
HRI	Human-robot Interaction
ID	In-distribution
IL	Imitation Learning
LiDAR	Light Detection and Ranging

MDP	Markov Decision Process
MPC	Model Predictive Control
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NLL	Negative Log Likelihood
NNT	Normalized Navigation Time
PBRL	Preference-based RL
PT	Path Tracking
OOD	Out-of-distribution
ORCA	Optimal Reciprocal Collision Avoidance
SAC	Soft Actor-Critic
SCRR	Safety-corridor Raise Rate
SFM	Social Forces Model
SHRR	Safety-human Raise Rate
SOTA	State of the art
SR	Success Rate
T&R	Teach and Repeat
UTD	Update-to-data
VLA	Vision-language-action Model
VLM	Vision-language Model

Chapter 1

Introduction

Achieving reliable robotic navigation remains a major challenge in deploying mobile robots into real-world environments. Applications such as food delivery, equipment transport, and warehouse logistics stand to benefit significantly from the integration of autonomous robots from manufacturer to end user. However, navigating dense human environments cluttered with static obstacles remains a substantial obstacle.

Social robot navigation research aims to enable robots to move safely and efficiently around humans. The primary challenge to social robot navigation is the ability to understand and predict a human's intent given the influence of the robot, the other humans, and the static obstacles in the scene. Traditional, learning-free approaches to this problem fall into three broad categories: reactive, decoupled, and coupled planning. Reactive methods tend to model humans as static obstacles [21]. While simple, computationally efficient, and explainable, these approaches are shortsighted because they do not anticipate human motion, often resulting in awkward or disruptive robot behavior [57, 80].

Decoupled planning addresses this limitation in reactive approaches by forecasting future human trajectories to produce cost maps for navigation [53, 80]. However, these methods assume that human motion is unaffected by the robot, ignoring the inherent interaction between humans and robots. This oversight can lead to problematic behaviors such as the robot 'freezing', when all possible robot paths intersect with the forecasted human trajectories [28]. Such an issue can arise even if all future human predictions are perfect because decoupled methods do not account for the interplay between the robot and the humans [82].

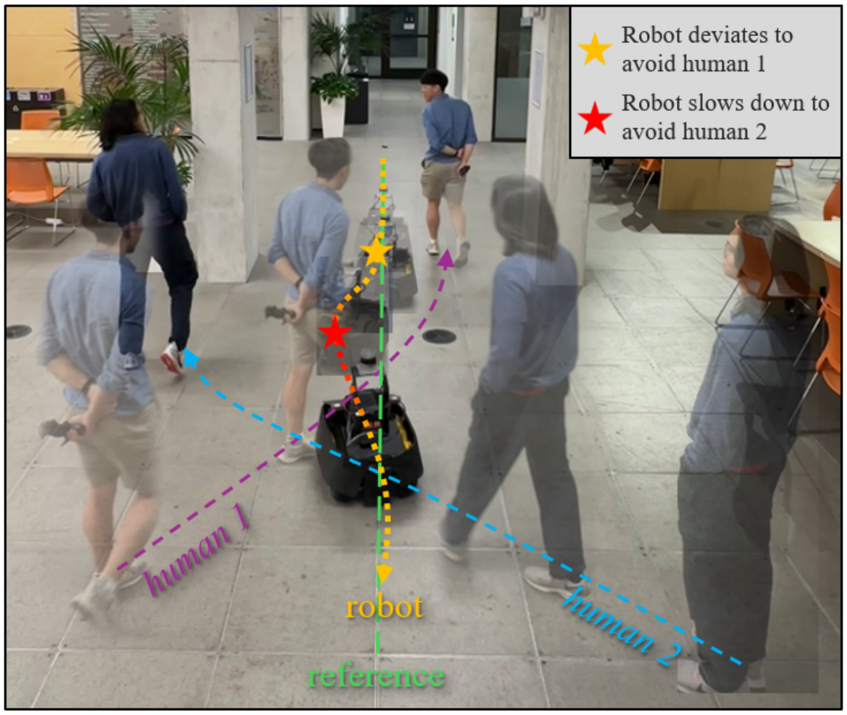
★ Robot deviates to avoid human 1
★ Robot slows down to avoid human 2
human 1
human 2
robot
reference

Figure 1.1: DR-MPC Navigating in the Real World. In this illustration, the robot deviates from its path to allow human 1 to pass and then slows down (red means slower speed) to let human 2 to pass before returning to its path.

Coupled planning approaches aim to resolve this by jointly optimizing the future motion of both humans and robots [68]. While theoretically promising, these methods are highly sensitive to inaccuracies in the human motion model. Whether learned or hand-crafted, building an accurate and generalizable human behavior model is notoriously difficult [68]. Furthermore, current human models typically rely on basic features and fail to incorporate higher-level reasoning about scene semantics—such as whether a human is on their phone or engaged in conversation with a friend.

Deep Reinforcement Learning (DRL) offers a compelling alternative by learning safe and efficient navigation policies directly from interaction data, thereby obviating the need for an explicit human motion model. The tradeoff is that DRL agents are typically trained in simulation because randomly initialized DRL agents are often unsafe and because the DRL training process if known to be data-intensive [18]. However, the use of simulation introduces a new challenge: simulators often rely on simplified and inaccurate human models, making the resulting DRL policies less effective in the real world. For example, the popular CrowdNav simulator models humans using Optimal Reciprocal Collision Avoidance (ORCA) [84], which assumes cooperative, deterministic human behavior [5]; but, humans in the real-world have varying levels of aggression, varying levels of attention to the scene around them, and varying reactions to the same situation. So, one consequence of using cooperative human models, such as ORCA, in simulation is that it can lead to overly aggressive robot policies that exploit human cooperation to reach the robot's goal more quickly [50]. Also, many simulators also neglect static obstacles entirely, further widening the sim-to-real gap [5]. To this end, we contribute DR-MPC: a DRL model that leverages Model Predictive Control (MPC) for path tracking while learning the human-avoidance component of the agent, enabling direct real-world training of a DRL agent and achieving proficient performance (Figure 1.1).

Another major limitation in DRL approaches to social navigation is the reliance on handcrafted reward functions. Most methods define rewards based on heuristic objectives such as goal progress, collision avoidance, and maintaining a socially acceptable distance from humans. However, this presents several challenges. First, reward tuning is non-trivial: for example, setting a high penalty for collisions can discourage exploration or cause the robot to remain stationary to reduce the risk of collision, thereby stalling the learning process. Second, these heuristic rewards can be inaccurate or overly simplistic. In scenarios such as narrow hallways, getting physically close to a person may be socially acceptable—or even necessary—yet such behavior would be penalized under rigid distance-based reward terms that numerous papers use [4, 6]. Finally, many subtle aspects of human comfort and social appropriateness—such as not cutting between groups or yielding when appropriate—are difficult to quantify or encode as explicit reward functions [22]. As a result, Imitation Learning (IL) offers an attractive alternative by learning from human demonstrations rather than manually specifying desired behavior. While several IL-based methods have been proposed and tried, the field still lacks a comprehensive evaluation in dense, real-world environments. For example, [41] evaluates in single-human settings or large open spaces. Further work is needed to robustly demonstrate the effectiveness of the IL approach for social robot navigation in complex and dynamic scenarios.

Even if the two prior limitations—simulation fidelity and reward function design—are addressed, reliably deploying DRL policies for social navigation in the wild remains a formidable challenge. As an overview (with more detail in Section 2.5), many existing methods use low-resolution or narrow field-of-view (FOV) sensors [19], require the robot and the humans to operate at reduced speeds [49], or require structured environments with VICON motion capture systems for human tracking [68] or localization [75]. Others introduce additional engineering overhead, such as manual camera-LiDAR calibration [4] or depth estimation via auxiliary networks [89], which either hinder deployment scalability or introduce perceptual uncertainty. Furthermore, real-time responsiveness is often lacking, with many systems exhibiting noticeable reaction delays in practice [6]. These limitations underscore the challenge of developing mobile robotic systems that are not only effective, but also robust, responsive, and practical for real-world, unstructured environments.

Thesis Structure. This thesis is organized as follows. We begin by presenting the core background and related work that frame the scope and motivation of our research. In Chapter 3, we introduce our novel decision process formulation, which unifies path tracking with human avoidance—in contrast to most DRL-based social robot navigation works that neglect static obstacles entirely. In the same chapter, we describe our implementation of this environment in both simulation and hardware; the hardware setup itself is a systems contribution, relying on a single sensor to enable reliable and efficient operation. Chapter 4 then presents our original sample-efficient DRL algorithm: Deep Residual Model Predictive Control (DR-MPC) for real-world social robot navigation [32]. DR-MPC integrates Model Predictive Control (MPC) for path tracking to accelerate policy learning. We demonstrate the effectiveness of this integration in both simulation and the real world, showing that DR-MPC can be trained directly in the real world with less than four hours of data. In Chapter 5, we extend current Imitation Learning (IL) approaches for social robot navigation to real-world, 'in-the-wild' scenarios. Specifically, we show that vanilla Behavioral Cloning (BC) is insufficient. We contribute several key design choices in the training algorithm and model architecture are required to achieve proficient closed-loop performance; we experimentally support these design choices. Finally, we conclude with a discussion of future work and outline promising directions for DRL and IL in social robot navigation.

Chapter 2

Background and Related Works

Fundamental concepts and the approaches that are used and referenced throughout this thesis are stated in this chapter. Specific technical literature related to a project is placed in that project's section.

2.1 Reinforcement Learning and Imitation Learning

In this section, we provide an overview of the value propositions of Deep Reinforcement Learning (DRL) and Imitation Learning (IL) pertinent to social robot navigation. A visual representation contrasting the two methods can be found in Figure 2.1. Rather than focusing on technical derivations, we highlight conceptual differences and practical considerations. Core references for the DRL and IL material used for this thesis include: RL fundamentals [77], Soft Actor-Critic (SAC) [29,30], Discrete SAC (DSAC) [12], Behavioral Cloning (BC) [65].

2.1.1 Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning (DRL) is a powerful learning framework that optimizes sequential decision-making under uncertainty. In the context of social navigation, the robot must choose an action that factors in not only its own objectives but also the behavior of nearby humans, which the robot can not directly control. Thus, the robot's current action affects future observations, which in turn determine the rewards it accumulates. The uncertainty in this

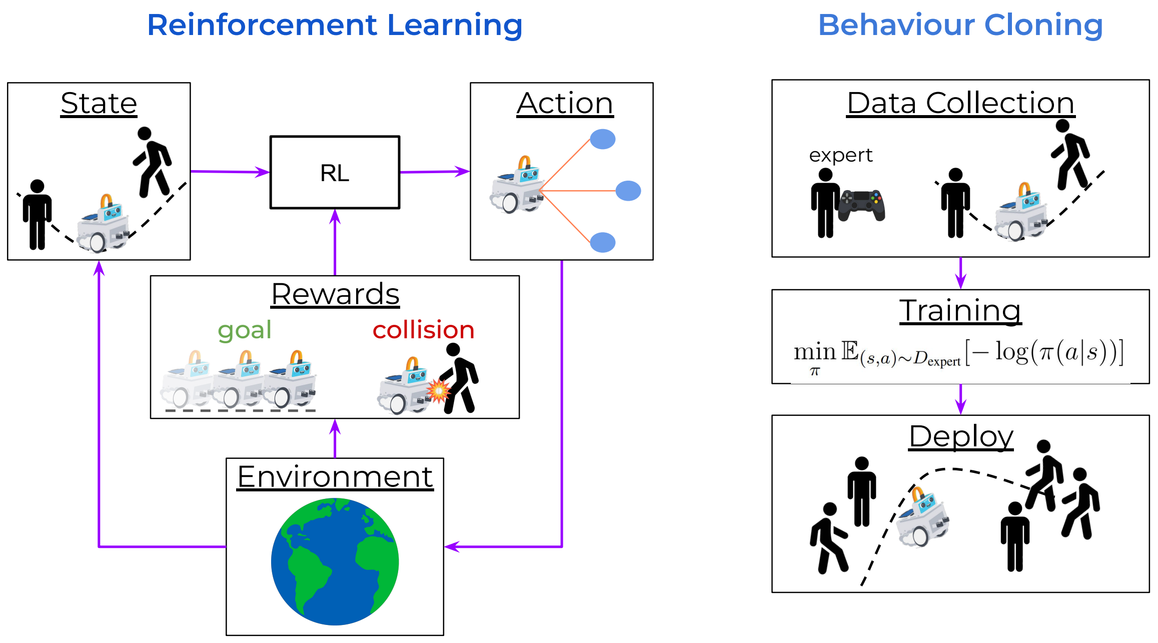

Figure 2.1: RL vs BC. This figure visually contrasts the learning process between RL and BC. The left side depicts the RL training process, which involves repeatedly interacting with the environment. The right side shows BC operating in three distinct phases of data collection, training, and deployment.

problem stems from the humans' reactions to the robot.

More formally, the robot follows a policy π, which induces a distribution $p_\pi(\tau)$ over state-action trajectories τ. The goal is to find the optimal policy π^* that maximizes the expected cumulative reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{p_\pi(\tau)} \left[\sum_{t=1}^{T} r(s_t, a_t) \right] \tag{2.1}$$

This objective accounts for the stochasticity in human motion because the environment dynamics are embedded in $p_\pi(\tau)$. Importantly, model-free methods such as SAC do not require explicit modeling of the human transition dynamics, which are embedded in $p(s_{t+1}|s_t, a_t)$. Instead, the algorithm learns solely from sampled transitions collected during interaction with the environment. This is particularly appealing in human-robot interaction settings where accurate human modeling is very difficult. Another note is that while DRL typically requires the policy to be differentiable, the reward function itself does not need to be. Reward signals can be either hand-engineered or learned from data, including through approaches such as Preference-Based

Reinforcement Learning (PBRL) or Inverse Reinforcement Learning (IRL). In PBRL, the reward is inferred from human preferences or rankings rather than explicit numerical feedback. On the other hand, IRL attempts to recover the reward function from expert demonstrations.

2.1.2 Imitation Learning (IL)

Reward design is a critical component of DRL methods, and heuristics have been used to attempt to better design rewards for better qualitative navigation [69]. Imitation Learning offers an alternative that avoids reward design and reward learning entirely. Instead of optimizing a cumulative return, IL learns directly from expert demonstrations. Given a dataset D_{expert} of state-action pairs (s, a^*) generated by an expert, which is typically generated by a human driver, the goal is to train a policy π to mimic the expert. Behavioural cloning (BC) is the most prominent IL method to match the policy's behaviour with the expert's behaviour. In the case of discrete actions, BC typically uses the cross-entropy (CE) loss, which is equivalent to minimizing the negative log-likelihood (NLL) of the expert's actions under the policy's distribution. :

$$L_{\text{discrete}} = \mathbb{E}_{(s,a)\sim D_{\text{expert}}}[-\log(\pi(a|s))] \tag{2.2}$$

In BC with continuous action spaces, we typically minimize the mean squared error (MSE) between the predicted and expert actions:

$$L_{\text{continuous}} = \mathbb{E}_{(s,a)\sim D_{\text{expert}}}[||\pi(a|s) - a||_2^2] \tag{2.3}$$

In this framework, the task objectives are encoded implicitly within the demonstrated behavior. The desired outcome is that if the learned policy can reliably reproduce expert-like actions across encountered states, the resulting robot behavior will resemble that of the expert. Generally, we require the expert demonstrations to be near optimal because BC on its own has no systematic way to improve its closed-loop performance. One important note in these BC loss functions is that we do not require modeling the human because the dataset is generated through interaction with the environment.

One of the key open challenges in IL is effective policy representation, especially when expert behavior is multi-modal. This issue is particularly relevant for social robot navigation, where multiple valid trajectories often exist in a given scenario. For instance, if a person is standing in the middle of the path, a robot could reasonably go around them on the left, on the right, or wait to see if the person moves. Another reason social robot navigation exhibits multi-modal behavior is the variability in expert demonstrations, both across different experts giving demonstrations and within the same individual over time. This variation arises from personal preferences and occasional human inconsistency. As a result, it is prudent to use policy representations that can capture a diverse set of socially appropriate behaviors.

BC methods that learn policies by regressing directly to continuous actions can struggle in such settings. When the same state corresponds to multiple valid expert actions, minimizing mean squared error (Equation 2.3) tends to produce an averaged action. In the example of a human standing in the middle of the path, if the expert demonstrations only include going left or right, the policy might learn to drive straight ahead—an unsafe behavior that could result in a collision. Discretizing the action space and optimizing with the CE loss (Equation 2.2) can better handle multi-modal behavior by allowing the policy to assign probability to multiple distinct actions. The tradeoff, however, is reduced action precision due to coarse discretization. More expressive approaches such as Implicit Behavioral Cloning (IBC) [20] learn an energy function over actions and sample from it, capturing complex and discontinuous distributions. Similarly, diffusion-based methods [11] model expert behavior through a generative denoising process, naturally accommodating multi-modality during sampling. Given the inherently multi-modal nature of expert demonstrations in social robot navigation, it is important for both the model architecture and the policy's action space to explicitly account for this characteristic.

2.1.3 Comparison and Practical Tradeoffs

Choosing between DRL and IL depends on a number of factors. IL is generally preferred when the reward function is difficult to specify—especially in tasks involving nuanced human preferences or social norms. However, IL requires expert data, which is often expensive to collect in terms of time and effort. The robot is often manually teleoperated to collect the data. One way to autonomously collect data for IL model development is to simplify the scenario so that an expert policy can be simplified; we later use this methodology to explore design choices.

By contrast, DRL—especially in simulation—enables fully autonomous data collection. Failed episodes can be automatically reset, and the policy can iteratively improve through trial and error without human supervision. Although this automated procedure is possible in simulation, in the real world, careful supervision is required and human effort is often required to reset the episode.

Both DRL and IL have proven useful in social navigation contexts, where modeling human behavior is challenging and interaction dynamics are complex. Their ability to jointly reason over robot and human actions—either through sampled rollouts or demonstration data—makes them powerful tools for building socially competent robot navigation systems.

2.2 Social Navigation Simulators

Several 2D and 3D simulators have been and continue to be developed for social robot navigation. Simulators offer a way to safely test and experiment with different approaches. Simulators also offer the possibility, in theory, of developing "sim-to-real" approaches, which can substantially reduce the amount of real-world learning and mistakes that robots make. A popular simulator for social navigation is the 2D CrowdNav simulator [5], designed for waypoint navigation in open spaces with the human policy defined by ORCA [84]. The ORCA model is heavily used in literature to represent the human policy because of the guarantee: if all agents in the scene follow ORCA, no collisions will occur. CrowdNav has facilitated substantial DRL-based social robot navigation model development [8, 19, 49–51, 56, 89, 96, 99].

Unfortunately, the cooperative nature of the ORCA policy creates a large sim-to-real gap because humans in the real world have varying levels of cooperation and exhibit cooperation in different ways. Prior work has shown that training a DRL policy in environments where humans follow ORCA and that the robot is visible to the humans often leads to dangerously aggressive behavior: the robot learns to push humans aside to reach its goal [50]. This behaviour is also a result of the difficultly of reward tuning, where weighing the goal reward too heavily will cause more aggressive behaviour. Consequently, the *invisible testbed*—when the robot is invisible to the humans—was adopted as a benchmark standard [49, 50, 89]. Although the invisible testbed reduces the DRL agent's aggression because the robot can no longer force the human modeled with ORCA out of the way, the problem becomes equivalent to decoupled planning where the DRL agent does not learn how its action will influence humans [50]; if the robot can simply accurately predict future human motion, the robot can succeed at navigation.

Looking beyond CrowdNav, other simulators use alternative human motion models, such as variations and combinations of the Social Forces Model (SFM) and behaviour graphs to emulate more realistic human behaviour [42, 83, 92]. Unfortunately, even the latest simulators struggle with human realism and often exhibit unsmooth human motions and enter deadlock scenarios. The reason for these behaviours is because models such as ORCA and SFM are reactive controllers and only consider the current timestep: the notion of waiting for others to pass and not progressing to my own goal at this moment does not exist, which is why deadlocks can occur. These deficiencies create a significant sim-to-real gap, where models trained in simulation often perform differently—–and usually worse—–when applied in the real world [97]. While these simulators have been essential tools for advancing DRL model architectures, the most accurate data for DRL social navigation is real-world data. So, sample-efficient DRL algorithms are valuable because it is then possible to directly train the DRL policy in the real world, thus avoiding unnecessary inductive biases introduced by simulators.

Although there has been a shift from 2D simulators such as CrowdNav to more visually rich 3D simulators such as [42] and [92], the realism of 3D human motion remains limited. Coordination between navigation algorithms such as ORCA or SFM and human animation is often hard-coded. For example, when a human navigates toward a goal using ORCA—which outputs actions in the form of (v_x, v_y)—the simulator simply plays a walking animation. This can lead to unrealistic motion, such as a character pivoting abruptly on one foot during a sharp turn, which does not resemble natural human movement. Another issue lies in the abrupt transitions between behavior animations. For instance, a human walking normally may suddenly stop, face the robot, and raise their hands in fear once the robot reaches a certain distance, as seen in [66]. These hard-coded behaviors fail to capture the fluidity and subtlety of real human reactions. This limitation has significant implications: it makes it difficult to develop models in simulation that rely on more than just the human's (x, y) position. If a model aims to leverage richer cues—such as walking gait or skeletal posture—for improved predictive power, current simulation environments fall short. The lack of realistic and dynamically coherent human motion makes simulation an inadequate tool for such tasks.

2.3 Deep Reinforcement Learning Social Navigation Models

The field of DRL social navigation, also commonly referred to as crowd navigation, focuses on enabling a robot to navigate among humans in open environments. Over the past decade, significant progress has been made in incorporating Machine Learning (ML) advancements to enhance DRL social navigation model architectures. These architectures primarily aim to reason about crowd dynamics in both spatial and temporal dimensions while ensuring the model is invariant to the order of human inputs. The permutation invariance to humans is a unique consideration for the ML models: given a set of humans, the output of the model should be the same regardless of the order of the sequence.

Early models considered pairwise comparisons between the robot and each human, using sequence-invariant operators such as maximum and minimum [9, 10]. Another early approach defined an ordering convention based on the distance of each human to the robot [19]; however, such heuristic definitions of importance are prone to failure in edge cases. For instance, a human who is farther away but walking toward the robot is likely more relevant than a closer human who is walking away from the robot.

Consequently, a major shift in model architectures occurred with the introduction of attention mechanisms to learn the relative importance of each human [1, 5, 96]. These mechanisms enable the robot to generate a crowd embedding for decision-making based on learned attention scores. This design is advantageous because attention allows the model to reason about both robot-human and human-human interactions.

Concurrently, Graph Neural Networks (GNNs) emerged as a popular alternative [1, 4, 49]. Representing a crowd as a graph is a natural formulation, and message-passing algorithms within GNNs facilitate complex relational feature learning.

Building on these models, recent advancements have incorporated temporal elements through spatio-temporal graphs and incorporating trajectory histories of both humans and the robot into the attention mechanisms [50, 89], enabling reasoning at the trajectory level rather than at individual time steps.

One major limitation of a purely ML–driven model design is the lack of sample efficiency in the resulting DRL algorithms. While the absence of inductive biases allows the model to learn from data without constraints, it also means that training requires substantial amounts of interaction data—typically limiting learning to simulation. A DRL agent initialized with random behavior must explore extensively, often relying on chance to discover successful strategies before it can begin exploiting them effectively. In DR-MPC (Chapter 4), we demonstrate that learning can be restricted to the human avoidance component, while providing a strong inductive bias for path tracking through MPC. This leads to significantly faster training and strong performance, both in simulation and in the real world.

Latest developments in the past couple of years attempt to leverage foundation models for navigation [58, 87, 88]. These models overcome the human permutation invariance problem by semantically reasoning about the scene. These models can also enable zero-shot navigation; however there are tradeoffs. The foundation model can semantically reason about the scene but can not geometrically analyze the situation to provide precise geometric paths or velocity commands.

In our thesis, we do not focus on developing a novel architecture for traditional waypoint-based crowd navigation. Instead, we leverage SOTA architectures for processing human-related inputs and incorporate them into our work.

2.4 Imitation Learning for Social Robot Navigation

Compared to DRL-based approaches, fewer works explore IL for social robot navigation. The primary reason is the need for expert demonstrations, which are costly to obtain—typically often requiring teleoperation of the robot. Surprisingly, few studies explore IL entirely in simulation [94]. One hypothesis is that if one is already teleoperating an agent for data collection in simulation, one might as well collect real-world data directly to avoid duplicating effort.

2.4.1 Behaviour Cloning for Social Robot Navigation

A straightforward approach to IL to social robot navigation is to use BC, where a model is trained via supervised learning to map observed states to expert actions. Several works have adopted BC for social robot navigation [31, 41, 64, 67, 78], differing in their choice of datasets, robot platforms, network architectures, and action representations. For example, [78] performs BC directly from raw depth images, while [41] processes 2D LiDAR scans into bird's-eye view (BEV) images. Given that both inputs are image-based, most of these methods rely on Convolutional Neural Networks (CNNs) as the primary feature extractor. However, the models differ in how they fuse additional contextual information, such as the robot's goal pose or recent trajectory. The output representation also varies: [67] predicts a sequence of future 2D trajectory waypoints, whereas [41] directly regresses low-level velocity commands for the robot. Despite the variation in design choices across these works, such as input modality, network architecture, and action representation, there is no clear consensus on a universally superior approach, as most approaches do not benchmark against each other because each paper's application is unique, from the sensors on the robot to the testing environment.

An exception that does perform useful ablations is [94], which conducts all experiments in simulation—enabling more controlled benchmarking and evaluation. They use the Dynamic Window Approach (DWA) as an expert planner to generate demonstration trajectories within the PEDSIM crowd simulation environment [27]. To ensure demonstration quality, trajectories involving collisions are filtered out, an approach commonly referred to as Filtered Imitation Learning. For input representation, they transform raw LiDAR point clouds into two separate occupancy grids: one encoding pedestrian velocity in the robot's x-direction and the other in the y-direction, with empty cells set to zero. This processed representation of dynamic agents was found to improve policy performance, likely due to improved sample efficiency compared to learning directly from raw sensory input. As we later discuss in Section 2.5, limitations in real-world sensing pipelines and hardware often motivate end-to-end learning from raw inputs to control commands; however, this work suggests that learning from more structured intermediate representations can be beneficial when possible.

Most datasets used in these works were collected in real-world settings and span between one and ten hours of demonstration data. However, the resulting policies are often evaluated in constrained scenarios with limited human interaction [35, 41]. Consequently, while IL can theoretically encode socially compliant behavior from demonstrations, its practical ability to generalize socially appropriate navigation in diverse real-world settings is not well established in the literature.

We motivated earlier that social navigation inherently involves multi-modal behaviors; however, many of the aforementioned BC approaches rely on regression with a MSE loss, which implicitly assumes a unimodal Gaussian distribution over actions, which as we previously discussed may produce unsafe actions. To address this, [31] explored discretizing the heading direction in the action space, allowing the model to represent multiple valid action modes, which led to improved performance.

2.4.2 IL Leveraging Existing Pedestrian Datasets

Several works aim to leverage larger datasets originally intended for pedestrian trajectory forecasting. For example, [48] uses the New York Grand Central dataset, containing 12,000 human trajectories, and trains a policy using Generative Adversarial Imitation Learning (GAIL) [36]. While their results qualitatively demonstrate socially compliant trajectory prediction, the policy is not evaluated in a closed-loop control setting where the robot physically navigates among humans.

Similarly, [31] uses the ETH pedestrian dataset [63] to train a model that outputs instantaneous speed and direction commands rather than full future trajectories. They evaluate their method in the invisible testbed, where a robot path is simulated from a start to a goal location, and the trajectory is deemed successful if it avoids collisions with the pre-recorded human trajectories. Importantly, because the humans in the dataset cannot react to the robot, this setup lacks interaction and tests only passive compliance.

More general-purpose datasets for social robot navigation are beginning to emerge. [59], for example, collected over 20 hours of human navigation data using a head-mounted sensor suite. The demonstrator walked in a socially compliant manner, aiming to enable cross-embodiment learning where models trained on human-centered data may generalize to robotic navigation tasks; however, this cross-embodiment learning was not done.

2.4.3 Other Applications of IL for Social Robot Navigation

IL and supervised learning have been used in other facets of social robot navigation. For example, IL is sometimes used in support of DRL for social navigation. In some works, IL is used to pre-train a DRL policy before fine-tuning with DRL [4,6]. These expert demonstrations are typically generated using the ORCA algorithm. However, because of the use of ORCA, this method of pre-taining does not apply to robots with differential-drive kinematics.

In [93], GAIL is used to learn an expert state-action distribution, which is then incorporated as part of the reward function for DRL. This hybridization allows the agent to benefit from expert demonstrations while still refining its behavior through exploration based on other heuristic-designed rewards.

Although not designed for dense social navigation, [13] applies BC to learn navigation commands based on human gestures. By interpreting arm gestures captured from camera images, the robot can respond appropriately, such as turning left or right when prompted.

Hirose et al. propose a supervised learning approach that forecasts human motion as a function of the robot's intended future actions [35]. Their system searches for an action sequence that not only optimizes the robot's navigation objectives but also minimizes the divergence between predicted pedestrian trajectories under robot motion versus no robot motion (i.e., staying still). This human-centric optimization framework reflects a growing interest in minimizing social disturbance or discomfort [22,56], marking a promising direction for future IL-based social navigation methods.

Finally, recent works have begun leveraging vision-language models (VLMs) for social navigation, using IL as a distillation mechanism to train deployable policies [34, 58]. Since VLMs are large and computationally intensive, they are not well-suited for real-time inference on robotic platforms. Instead, these approaches perform offline processing using VLMs to extract expert actions or score trajectories from diverse data sources. The resulting supervision is then used to train smaller, lightweight policies capable of efficient real-time execution while retaining the VLM's social reasoning capabilities.

2.5 Real-world Social Navigation Deployment

There is currently no de facto hardware platform for real-world deployment of socially aware robot navigation, so prior work rely on different hardware systems with varying capabilities. Although many algorithms perform well in simulation, building a physical system that robustly handles social scenarios remains a major open challenge. From a systems perspective, real-world deployment can be viewed as a constrained optimization problem balancing five primary objectives: (1) maximizing the robot's field-of-view (FOV), (2) improving the quality of human detection and tracking, (3) reducing inference latency, (4) minimizing hardware and operational cost, and (5) ensuring ease of integration and setup. Each of these objectives is associated with critical constraints—for instance, insufficient FOV or unreliable human detection leads to unavoidable unsafe behavior, while excessive hardware cost or complexity hampers adoption and reproducibility.

Many existing systems fail to simultaneously satisfy these constraints, limiting their real-world applicability. Sensor FOV is often constrained: [4, 6, 50, 58, 75, 79, 89] rely on narrow-angle sensors such as a single camera, making it impossible to construct potentially important parts of the scene for navigation. Human detection quality is also compromised in systems using 2D LiDAR human detection methods [19, 49]. Some methods such as [89] rely on pre-trained depth estimation networks to extract the real-world positions of the humans, which is significantly less reliable than LiDAR for accurate human localization.

In addition to perception, many approaches have limitations in computational efficiency. For example, [50, 58] offloads policy inference to a remote server, introducing network latency and requiring a stable connection, which is impractical in outdoor environments. Others such as [4, 6, 49, 58, 75, 79, 89] require humans to move slowly to accommodate limited robot speeds and slow inference pipelines. Such constraints greatly limit the deployment of such social robot navigation methods in real-world environments.

The setup complexity of these systems further restricts their scalability. Multiple systems [68, 75] depend on VICON motion capture systems for either robot localization or human tracking—an impractical requirement in real-world venues such as shopping malls, airports, or hospitals. Others rely on multi-sensor fusion, combining LiDAR and camera data [4, 19], which necessitates careful and often brittle calibration procedures. Even slight misalignments between sensor extrinsics can cause large errors in human localization, making real-world deployment error-prone and labor-intensive.

Ultimately, designing a real-world social navigation system that is safe, fast, accurate, and easy to set up remains a significant open challenge. The ideal system would need to generalize across diverse environments and user behaviors without relying on specialized infrastructure, all while running efficiently on onboard compute.

Furthermore, these real-world considerations pose significant challenges for sim-to-real transfer. Most simulation environments fail to capture critical deployment constraints such as limited field-of-view, latency, sensor calibration errors, and imperfect human detection. While some simulators attempt to emulate constrained FOV [49, 50], other limitations—such as compute bottlenecks, sensor noise, and multi-sensor synchronization—are rarely modeled. Another overlooked factor is the delay in state generation for the policy: generating the point cloud, performing human detection and tracking, and executing the model all introduce latency. This contrasts with simulation, where the environment is typically paused between timesteps to allow instantaneous policy inference. Such discrepancies make it difficult to directly transfer policies trained in simulation to real-world systems without extensive fine-tuning or additional engineering workarounds.

2.6 Teach and Repeat

Teach and Repeat (T&R) is a foundation for autonomous mobile robotic deployment, especially in environments where GPS is unavailable or unreliable [24]. From a practical standpoint, T&R enables a robot to retrace paths demonstrated by a human operator by leveraging sensor data collected during a "teach" phase to localize and thus "repeat" the path during subsequent missions. Over the past decade, the Visual Teach and Repeat (VT&R) framework has matured through several iterations, culminating in the modern and modular VT&R3 system [24]. VT&R3 integrates a flexible architecture that supports a variety of sensor configurations, including stereo cameras, LiDAR, and RADAR, making it highly suitable for long-range, infrastructure-independent navigation tasks.

Sehn et al. [71] introduced a LiDAR-based extension to VT&R3 that employs a sample-based motion planner in a curvilinear coordinate system. By penalizing both obstacle proximity and lateral deviation, the robot can generate smooth trajectories that safely navigate around static obstacles while minimizing the distance to the taught path. The next step towards robust T&R is to enable repeats in dynamic environments where humans may navigate alongside the robot. This multi-objective problem of path tracking with human avoidance is the core focus of this thesis. Our developed systems to achieve this goal build upon T&R.

Chapter 3

The Environment in Simulation and Hardware

In this chapter, we introduce our decision process formulation for social robot navigation, which combines path tracking and human avoidance. This thesis relies on two key components: a simulation environment and a hardware system. Simulation enables fast iteration and quantitative benchmarking, while the hardware setup demonstrates that the developed approach is valid and can be feasibly deployed for the real world, which is the ultimate goal for social robot navigation.

3.1 Decision Process Formulation

The prominent DRL social navigation MDP in [5] achieves two objectives: human avoidance and waypoint navigation. This formulation suffers from jerky waypoint transitions and does not account for static obstacles. We redefine the MDP to use path tracking with virtual corridors instead of waypoint navigation, addressing both issues. A visual of our decision process formulation is shown in Figure 3.1. We define the reference path and two bounding corridor paths—staying within these corridors implies safety from static obstacles. This representation generalizes to environments with any static obstacle configuration. For this thesis, we simplify the problem and reduce the state space dimensionality by assuming a constant corridor width. With a path representation, we can handle short and long navigation goals. Additionally, path

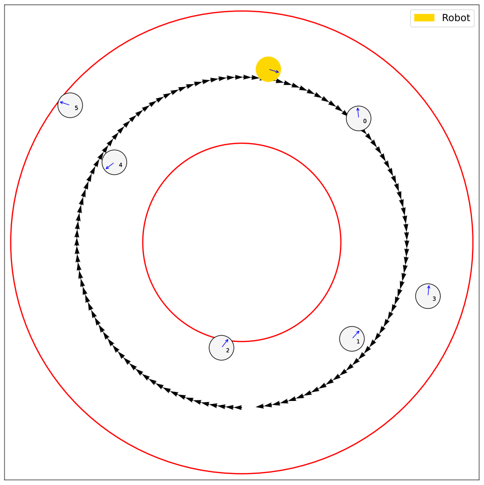

Figure 3.1: Our Simulator. A visual of our simulator where we combine path tracking within virtual corridors and human avoidance.

tracking reduces the amount our DRL model needs to learn, as global planning in a mapped environment is a mostly solved problem that provides the optimal navigation homotopy among typical static obstacles. Furthermore, by setting up the problem as path tracking, we can also leverage high-performing non-learning-based models such as MPC for further learning reduction. Lastly, practically speaking, we are able to leverage T&R for efficient real-world deployment on long navigation tasks.

3.1.1 State Space

Path tracking and human avoidance require independent sets of information; thus, we construct our state space $\mathcal{S} = \{\mathcal{S}_{\text{PT}}, \mathcal{S}_{\text{HA}}\}$, where $\mathcal{S}_{\text{PT}}$ is the state information for path tracking and $\mathcal{S}_{\text{HA}}$ is the state information for human avoidance.

As in [50], we exclude human velocities due to the difficulty of estimating these quantities in the real world and assume a constant human radius. The robot frame is defined with the x-axis aligned to the heading, the z-axis pointing upward (opposite gravity), and the y-axis following the right-hand rule. If at time t there are n_t visible humans: $\mathcal{S}_{\text{HA}} = \left\{\mathbf{v}^{t-H:t-1}, \mathbf{r}^{t-H:t}, \mathbf{q}_1^{t-H_1:t}, \ldots, \mathbf{q}_{n_t}^{t-H_{n_t}:t}\right\}$, where $\mathbf{v}^{t-H:t-1}$ is the robot's past commanded linear and angular velocities from time $t-H$ to $t-1$, $\mathbf{r}^{t-H:t}$ is the robot's past (x, y) positions in the current robot frame from time $t-H$ to t, and $\mathbf{q}_i^{t-H_i:t}$ is the i^{th} visible human's past (x, y) positions in the current robot frame from time $t-H_i$ to t, capped at length H. Standard DRL social navigation assumes a fully observable environment that results in no fluctuation for history length, but in the real world, human tracking is prone to identity switches and humans may enter or leave the scene, so fluctating history lengths is important to incorporate into the state space.

For path tracking, we select a local representation of the path similar to [37], which generalizes to any path length. Given that $\mathbf{p}_l$ is the closest localized node to the robot, $\mathcal{S}_{\text{PT}} = [\mathbf{p}_{l-L}, \ldots, \mathbf{p}_l, \ldots, \mathbf{p}_{l+F}]$, where L is the look-back and F is the look-forward parameter. These path nodes are also transformed into the current robot frame. Based on the robot's velocity limits, the discount factor, and the MDP's time step, we approximate the number of local path nodes to include in $\mathcal{S}_{\text{PT}}$ so that the cumulative reward approximation error is negligibly small. For example, given the discount factor, we first determine the effective time horizon beyond which contributions to the return are negligible; we then estimate the distance the robot can travel within this horizon and include the corresponding number of path nodes. While a fully observable MDP with a local path representation would require indexing path nodes relative to the global path to prevent state aliasing, this format does not generalize well to paths of varying lengths. A full path representation is also not ideal because sequence model computations would scale with path length. Thus, we opt for a local path representation in the current robot frame, accepting small approximation errors.

In our state construction, we exclude any notion of "time into episode" in $\mathcal{S}$. In path tracking and human avoidance, time should not influence the optimal action. Therefore, when the timeout termination condition is reached, we perform Partial-Experience Bootstrapping (PEB) as proposed by [61].

3.1.2 Actions

Our DRL agent interacts at the level of velocity. Our action space for a differential drive robot is a linear and angular velocity: $\mathbf{a} = (v, \omega)$.

3.1.3 Rewards

As in our state space, our rewards are divided into the two tasks for this multi-objective problem. Our reward function considers path advancement, path deviation, goal reaching, corridor collisions, small speeds, human collisions, and human disturbance. Due to the strong reliance on reward shaping in this problem, we manually tune hyperparameters to obtain qualitatively acceptable behavior.

$$r = r_{\text{pa}} + r_{\text{dev}} + r^*_{\text{goal}} + r^*_{\text{cor-col}} + r^*_{\text{act}} + r^*_{\text{hum-col}} + r_{\text{dist}}, \tag{3.1}$$

The path-tracking reward terms are defined as follows, where rewards marked with an asterisk are terminal rewards:

- **Path advancement:** $r_{\text{pa}} = 5\Delta s$, where Δs is the arclength progress along the path.
- **Deviation:** $r_{\text{dev}} = 0.5d_{\text{xy}} + 0.03|d_\theta|$, where d_{xy} is the Euclidean distance and d_θ is the angular offset from the closest point on the path.
- **Goal*:** $r^*_{\text{goal}} = -5$ if the heading difference exceeds a specified threshold, and 0 otherwise. This design intentionally omits a positive goal-reaching reward to discourage the agent from rushing toward the goal at the expense of precise path following.
- **Corridor collision*:** $r^*_{\text{cor-col}} = -10$ for colliding with the corridor.
- **Minimal actuation*:** $r^*_{\text{act}} = -20$ if the sum of the robot's past H speeds falls below a threshold. This term prevents the robot from adopting a stationary policy out of excessive caution (i.e., avoiding exploration to minimize collision risk). We acknowledge that in extremely dense environments, remaining still may indeed be optimal, in which case this reward can be removed. Thus, it serves primarily as a reward-shaping term to encourage reasonable movement during training in sparser environments.

For human avoidance, we align our rewards with the two most important principles of human-robot interaction (HRI): safety and comfort [22].

- **Human collision* (safety):** $r^*_{\text{hum-col}} = -15$ for colliding with a human.

- **Disturbance penalty (comfort):** $r_{\text{dist}} = -\sum_{i=1}^{n_t}(5.6|\Delta v_{\text{h}}^i| + 3.5|\Delta\theta_{\text{h}}^i|)$, which penalizes the robot for causing changes in a human's velocity (Δv_{h}^i) and heading direction ($\Delta\theta_{\text{h}}^i$). In [56], it was demonstrated that using this disturbance penalty, it is possible to train in the visible testbed—when the robot is visible to the humans—while still achieving satisfactory qualitative results.

  These velocity and heading changes are computed relative to the human's state at the previous time step. Although this method inherently penalizes situations where humans intentionally change direction (an event beyond the robot's control), over a large number of trials such willful changes introduce only a constant bias across all models, which does not affect the optimal behaviour since DRL maximizes the expected cumulative reward.

Lastly, we augment these base rewards with two safety layers. We introduce safety-collision and safety-corridor termination conditions, which are more conservative versions of the human-collision and corridor-collision penalties. For example, if the robot and human have a separation distance of 0.1m, the safety-collision condition is raised. We assign the same penalty to the robot when it triggers a safety violation. While a safety violation does not guarantee a collision with a human or static obstacle, it is highly likely, thus we are slightly reducing the robot's theoretical performance limit for safety benefits. Such safety measures are common in robotic operations [74].

3.2 Simulator

We build on the CrowdNav simulator from [49], which models humans as 2D circles; the humans follow the ORCA policy. CrowdNav is highly configurable, allowing us to specify parameters such as the number of humans, their initial positions, and goal locations.

We augment the simulator to support path tracking in place of waypoint navigation. Our modular path-tracking module supports custom paths, localization, and auxiliary signals, such as distance to the path and progress along the path, necessary for reward computation. We also implement an MPC-based path-tracking controller as described in [71]. A visual example of the simulator is shown in Figure 4.3, which depicts the primary paths used throughout this thesis: clockwise (CW) circle, counter-clockwise (CCW) circle, and straight paths. The black arrows along each path indicate the spacing of nodes used for S_{PT}, while the red lines represent the corridor boundaries that the robot is expected to stay within. The robot is shown in yellow, and the humans are shown in grey. These four paths were chosen to facilitate cyclic training, where the end of one path naturally leads to the start of the next—e.g., the end of path 1 connects to the start of path 2, continuing through to path 4, which loops back to path 1. We adopt this continuous training setup in the real world as well. Further details on the simulator's usage is provided in later sections.

3.3 Hardware System

The complete hardware system is illustrated in Figure 3.2, which shows the full pipeline from a single LiDAR sensor to the final robot action. The following subsections describe each component in detail.

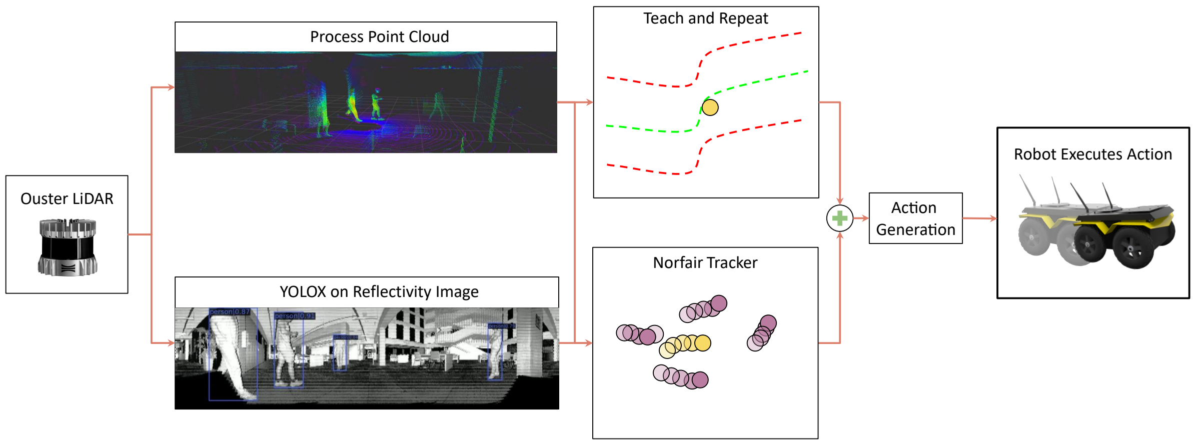

Figure 3.2: Full Hardware Pipeline. The Ouster OS0-128 LiDAR generates a detailed reflectivity image and a point cloud. The reflectivity image allows us to perform human tracking and the point cloud enables localization, path tracking, and depth recovery. With the state constructed from a single sensor, which we pass into an action-generation module, such as a DRL policy, which is then fed into the robot.

3.3.1 Robot and Sensor

As seen in Figure 3.3, our hardware platform is the Clearpath Robotics Jackal, equipped with an Ouster OS0-128 LiDAR. This sensor has a 90° vertical FOV and 360° horizontal FOV, providing both range and reflectivity data. The reflectivity image, shown in the top of Figure 3.4, is comparable to a low-resolution panoramic camera. By combining 2D computer vision models with LiDAR depth, we obtain 3D outputs without the need for a separate camera-LiDAR calibration—a step that adds complexity for real-world deployment.

Many camera-based systems used in prior real-world studies have horizontal FOVs under 90° [89], limiting their ability to detect nearby humans just outside the frame. In contrast, the OS0-128 captures a full panoramic view and operates at 10Hz, generating a full scan every 0.1s.

This robot can be smoothly teleoperated with a PS4 controller. We also use the PS4 controller to provide signals for data collection such as when an episode starts or ends.

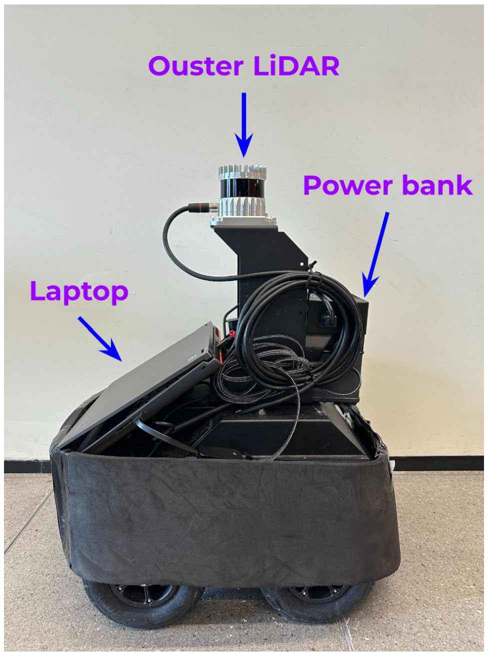

Figure 3.3: Mobile Robot Setup. The Clearpath Robotics Jackal is augmented with the Ouster OS0-128 LiDAR. We route data from the robot's onboard computer onto our external computer for faster processing. The robot's computer requires a 230W power bank to avoid power throttling.

3.3.2 Mapping, Localization, and Path Tracking

We use the open-source Teach and Repeat (T&R) codebase [25] to support mapping, localization, and path tracking. In LiDAR-based T&R, the robot is manually driven through an environment once, after which it can autonomously localize to the taught path and track it using MPC.

While computing MPC-based path-tracking commands, T&R also provides useful information for our learning framework (just like in simulation), including the localization index, path nodes transformed into the robot frame, and distance to the path—all of which are essential for constructing the state and computing the reward.

3.3.3 Human Tracking

Human detection is performed using a pre-trained YOLOX model from MMDetection, applied to both the original reflectivity image and a version shifted by $180°$ to capture wrap-around detections [7]. The only adjustments we applied to the reflectivity image were histogram equalization and rescaling. We merge detections from both images by selecting the detections closer to an image's center, as empirical observations indicate that detections of humans farther from the center exhibit greater flickering and instability. Using the depth image, we extract each human's 3D position relative to the robot from the center of the bounding box. These positions are then transformed into the world frame using T&R's localization output.

To track humans over time, we use the Norfair library [2], which implements a Kalman filter-based tracker in Euclidean space. Each new detection is used to update (correct) the track; between detections, the prediction step estimates the human's future position based on the human's past trajectory. Tracks can persist for several timesteps even if detections are missed, which is helpful for handling temporary occlusions or false negatives. In the worst-case scenario, a hallucinated human only results in conservative behavior.

We also experimented with image-space tracking followed by 3D extension. However, world-space tracking yielded more stable results. Occlusions are particularly problematic in image-space tracking. For instance, when two humans cross paths, image-space tracking often assigns bounding boxes incorrectly. While humans intuitively understand that the closer individual should retain their label, off-the-shelf image tracking models lack this spatial context.

3.3.4 Inference and Compute

Inference is run on a ThinkPad P16 Gen 1 with an Intel i7-12800HX processor and an NVIDIA RTX A4500 GPU. An Ethernet cable connects from the robot's onboard computer to our external laptop. Unfortunately, standalone laptop operation on battery suffers from power throttling due to insufficient power supply to both CPU and GPU. We resolve this with a 230W power bank. Neither the robot nor the laptop requires wifi for inference, so the entire system can operate in offline conditions.

While this setup supports inference, it cannot accommodate simultaneous DRL model training and inference without compromising real-time performance. In order to address this, we offload training to an external server over Wi-Fi. During each episode, we collect transition tuples (s, a, s', r) and send them to the server for training only after the episode concludes. At the same time, we pull the latest model weights from the server and then continue with the next episode of data collection. This asynchronous setup enables continuous online learning, albeit with some delay, as the model used to collect data is not yet trained on the most recent experiences.

This delay is primarily governed by the update-to-data (UTD) ratio. For example, a UTD of 3 indicates that the model is updated three times for every environment step. A higher UTD ratio increases the discrepancy between the model collecting data and the one being trained. While this lag is tolerable due to the off-policy nature of our DRL algorithm (SAC), it is still preferable for data to be collected by the most up-to-date model, as such data is more relevant for correcting Q-values and discovering new behaviors. To help mitigate this issue, we periodically pause data collection to allow training to catch up—a process conveniently aligned with robot and battery pack charging cycles.

Another crucial step for achieving real-time inference is applying TorchScript to the YOLOX model for human detection. YOLOX is a relatively large model (378MB), and running it in its raw PyTorch form through the mmDetection library on our setup takes an average of 106 ms for two images (the original and the wrapped-around image). After applying TorchScript, the inference time drops significantly to 45 ms.

A final key to achieving real-time performance is maximizing parallelism through multi-processing. Python's Global Interpreter Lock (GIL) allows only one thread to execute Python bytecode at a time per process, which limits the effectiveness of multithreading for CPU-bound tasks. As a result, even multi-threaded scripts may still execute on a single CPU core. To overcome this, we leverage Python's `multiprocessing` library and the `shared_memory` module, to parallelize computation across multiple processes while minimizing inter-process communication overhead.

This approach allows us to overlap computation and improve end-to-end system throughput. For example, while waiting for localization results from Teach & Repeat (T&R), we concurrently run YOLOX on the wrapped-around image in a separate process. We then manually synchronize results between processes using a nearest-neighbor matching scheme.

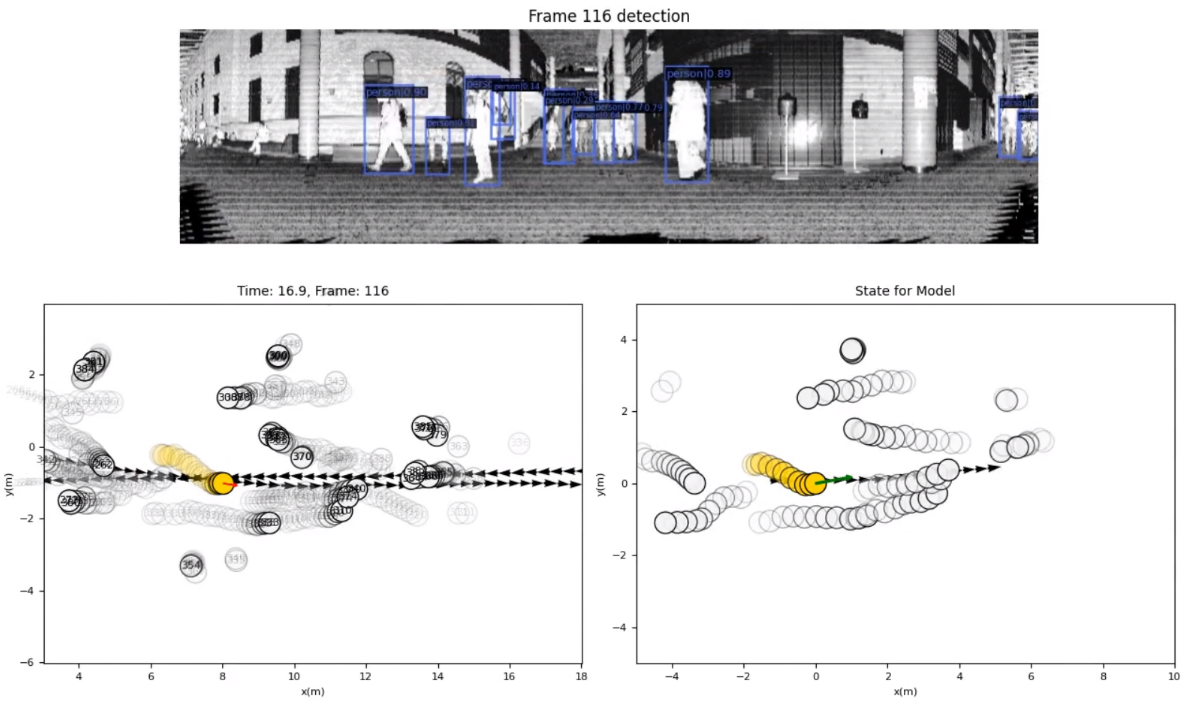

Figure 3.4: State Construction in the Real World. The top frame is the visualization of YOLOX's bounding boxes on the reflectivity image. The bottom-left frame is a visualization of the current state of the world including the path and humans; the numbers of the humans represents the tracker's ID. The bottom-right frame is the visual representation of the state that is passed to the robot's policy.

3.3.5 ROS System

The entire pipeline is implemented using ROS 2 [55]. For reproducibility, we highlight a subtle but important challenge related to the ROS 2 middleware. The Clearpath Jackal's onboard computer officially supports only Fast DDS, the default ROS 2 middleware. However, we discovered that the Ouster OS0 LiDAR requires the ROS 2 driver to run under Cyclone DDS in order to reliably operate at the full 10 Hz. Our finding has since been incorporated into the Ouster driver's documentation. Furthermore, any subscribers must also use Cyclone DDS to receive LiDAR topics at the correct rate.

This creates a DDS compatibility issue: our external computer cannot simultaneously subscribe to topics published using Fast DDS (from the Jackal) and Cyclone DDS (from the LiDAR). Through extensive testing, we identified a workaround: while a single ROS 2 node cannot subscribe across DDS implementations, a single *machine* can run multiple nodes under different DDS configurations. On the external computer, we subscribe to the Jackal's Fast DDS topics using one node and locally republish them. These republished topics are then accessible to Cyclone DDS-based nodes, including those that subscribe to the LiDAR topics—thus resolving the interoperability issue.

Chapter 4

Deep Residual Model Predictive Control

The publication associated with this section is “DR-MPC: Deep Residual Model Predictive Control for Real-world Social Navigation” [32].

4.1 Overview

Deep Reinforcement Learning (DRL) can learn navigation policies in environments with complex human motion. However, most prior work is limited to simulation, where human behaviors are often oversimplified and unrealistic. We propose Deep Residual Model Predictive Control (DR-MPC) to enable robots to safely and efficiently learn from real-world crowd navigation data.

DR-MPC combines model-free DRL with Model Predictive Control (MPC) to address two key challenges of DRL: poor initial performance and high sample complexity. Our approach is initialized with an MPC-based path-tracking controller and gradually learns to interact more effectively with humans over time. Crucially, we only introduce inductive bias for path tracking—an effectively solved problem under well-modeled, no-slip dynamics—while allowing the learned policy full flexibility in how it interacts with humans. To further improve safety and data efficiency, we incorporate an out-of-distribution (OOD) detection module that identifies risky states and steers the robot away from likely collisions.

In simulation, DR-MPC significantly outperforms prior DRL and residual DRL models. In real-world experiments, our method enables a robot to navigate crowded environments with few errors using under 4 hours of training data. A video of our approach can be found here: `https://youtu.be/GUZlGBk60uY`, and the code may be found here: `https://github.com/James-R-Han/DR-MPC`.

4.2 Related Works

Residual DRL is a key benchmark in this work. It integrates two policies: a user-supplied base controller and a learned residual policy [40]. The base controller can be rule-based, MPC-driven, or even a pre-trained policy. The DRL agent's objective is to learn corrective actions on top of this base controller. This formulation accelerates learning because, at initialization, the agent's expected action matches that of the base controller. If the base controller is competent, this leads to substantially better early performance compared to learning from scratch [40]. Moreover, the DRL agent typically operates in a reduced action space, focusing only on deviations from the base policy. This simplifies the exploration problem, further improving learning efficiency.

To the best of our knowledge, residual DRL has not been applied to social robot navigation, but some related works have attempted to incorporate classical control. For example, Kästner et al. [47] learn a DRL policy that switches between different controllers, each optimized for an individual task. Another work, Semnani et al. [72] switch from a DRL policy to a force-based model when close to humans. While these approaches do reduce the learning burden on DRL, they also limit the overall performance by the capabilities of the base controllers. In other words, when the overall policy selects a base controller, DRL can not optimize the performance of these individual controllers.

Motivated by these works, DR-MPC aims to fully exploit the capabilities of MPC path tracking. DR-MPC can either replicate the MPC action exactly or generate an action that diverges significantly from it. Unlike residual DRL, which follows the base controller actions in expectation, DR-MPC's initial behaviour almost exactly follows MPC path tracking, representing the best achievable performance without any prior information about humans. Additionally, DR-MPC dynamically learns when to integrate or disregard the MPC action, leading to significantly faster training compared to traditional residual DRL.

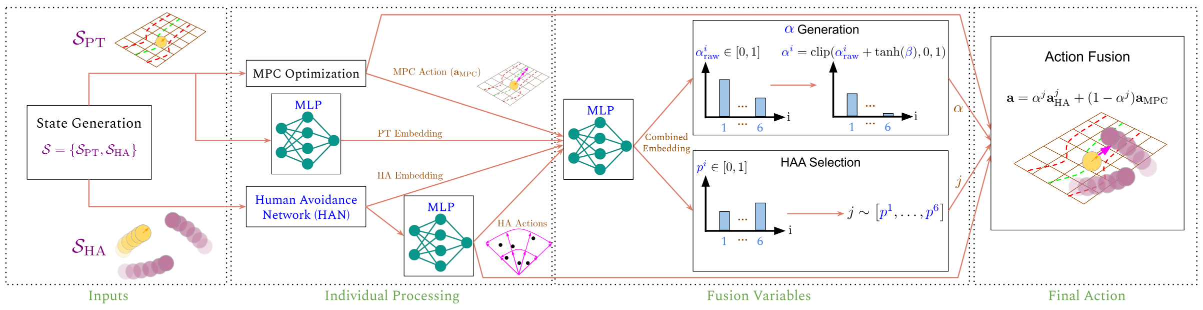
$\mathcal{S}_{\text{PT}}$
State Generation
$\mathcal{S} = \{\mathcal{S}_{\text{PT}}, \mathcal{S}_{\text{HA}}\}$
$\mathcal{S}_{\text{HA}}$
MPC Optimization
MLP
Human Avoidance Network (HAN)
MLP
MPC Action ($\mathbf{a}_{\text{MPC}}$)
PT Embedding
HA Embedding
HA Actions
MLP
Combined Embedding
α Generation
$\alpha^i_{\text{raw}} \in [0, 1]$
$\alpha^i = \text{clip}(\alpha^i_{\text{raw}} + \tanh(\beta), 0, 1)$
HAA Selection
$p^i \in [0, 1]$
$j \sim [p^1, \ldots, p^6]$
α
j
Action Fusion
$\mathbf{a} = \alpha^j \mathbf{a}^j_{\text{HA}} + (1 - \alpha^j)\mathbf{a}_{\text{MPC}}$
Inputs
Individual Processing
Fusion Variables
Final Action

Figure 4.1: DR-MPC Architecture. The dark blue text elements involve learning. From $\mathcal{S}_{\text{PT}}$, we generate the MPC path tracking action and a latent embedding of the path information using an MLP. From $\mathcal{S}_{\text{HA}}$, we use a SOTA human avoidance network to generate six actions for human avoidance. The model then fuses all the information to generate $\boldsymbol{\alpha}$ and $\mathbf{p}$, which generates the final action to maximize the human avoidance and path tracking rewards.

4.3 Policy Architecture

DR-MPC (Figure 4.1) consists of two main components: individual processing of $\mathcal{S}_{\text{PT}}$ and $\mathcal{S}_{\text{HA}}$, and the information fusion to generate a single action.

4.3.1 Path-tracking Embedding

We generate a latent embedding of the path using a Multi-layer Perceptron (MLP), and the MPC optimization from [71] generates the MPC path-tracking action $\mathbf{a}_{\text{MPC}}$.

4.3.2 Crowd Embedding

To process $\mathcal{S}_{\text{HA}}$, we modify [49] to handle varying-length human trajectories for off-policy learning. We refer to this adapted architecture as the Human Avoidance Network (HAN). The goal of HAN is to take varying number of humans with varying length position histories to output a crowd embedding that is permutation invariant to the order in which the humans are passed in.

Compared to [49], we use the humans' past trajectories rather than their forecasted trajectories. This way, DRL directly learns from the sensor noise embedded in the state, which will be important for real-world operation. If we had used a trajectory forecasting network like [49], the forecasting network would have inherited the errors from the sensor noise, preventing the DRL model from learning to handle this noise effectively.

For each human trajectory $\mathbf{q}_i^{t-H_i:t} = \left[\mathbf{p}_i^{t-H_i} \dots \mathbf{p}_i^t\right]$, we sequentially process it from time $t - H_i$ to t using a GRU to generate an embedded trajectory $\mathbf{e}_{\text{traj}}^i$. This step handles human trajectories of varying lengths, embedding them into a shared latent space. Next, with $\mathbf{E}_{\text{traj}} = \left[\mathbf{e}_{\text{traj}}^1 \dots \mathbf{e}_{\text{traj}}^{n_t}\right]$, we use three MLPs to generate the queries $\mathbf{Q}_{\text{traj}}$, keys $\mathbf{K}_{\text{traj}}$, and values $\mathbf{V}_{\text{traj}}$. We then apply multi-head attention using the scaled dot-product attention: $\text{MultiHead}(\mathbf{Q}_{\text{traj}}, \mathbf{K}_{\text{traj}}, \mathbf{V}_{\text{traj}})$ (see [85]). The output of this module is $\mathbf{E}_{\text{HH}}$, a $n_t \times d_{HH}$ tensor, where d_{HH} is the dimension of the human-human embeddings.

After reasoning about human-human interactions, we now need to reason about the robot-human interactions. So, we process the robot trajectory $\mathbf{r}^{t-H:t}$ by embedding it with an MLP into $\mathbf{e}_{\text{traj}}^{\text{robot}}$. We then compute the robot-human attention. Here, the keys $\mathbf{K}_{robot}$ are generated from $\mathbf{e}_{\text{traj}}^{\text{robot}}$ and the queries $\mathbf{Q}_{\text{HH}}$ and the values $\mathbf{V}_{\text{HH}}$ from $\mathbf{E}_{\text{HH}}$. The result of this multi-head attention network is the embedding $\mathbf{e}_{RH}$.

Finally, we concatenate $\mathbf{e}_{RH}$ with $\mathbf{v}^{t-H:t-1}$ and pass this tensor through one last MLP to obtain the crowd embedding $\mathbf{e}_{HA}$, which is then used to generate the 6 mean actions for human avoidance. Importantly, despite the number of humans in the scene or variations in their trajectory length, we produce a constant vector size for $\mathbf{e}_{HA}$ that encapsulates both spatial and temporal information from all the humans and the robot. This embedding $\mathbf{e}_{HA}$ is then used for downstream action generation and selection.

4.3.3 Action Learning

The crowd embedding generates six candidate human-avoidance actions ($\mathbf{a}_{\text{HA}}$), where each $\mathbf{a}_{\text{HA}}^{\text{i}}$ is positioned in a different cell of the action space. The human-avoidance action space is partitioned into six cells defined by two linear velocity bins $[v_{\text{min}}, v_{\text{middle}}]$, $[v_{\text{middle}}, v_{\text{max}}]$, and three angular velocity bins $[w_{\text{min}}, w_{\text{lower}}]$, $[\omega_{\text{lower}}, \omega_{\text{upper}}]$, $[\omega_{\text{upper}}, \omega_{\text{max}}]$. We include multiple actions because often several viable actions exist for human avoidance. We found empirically that this design of enforcing the human-avoidance actions to be different reduces learning time compared to having the model learn the diversity. We also found empirically that using six actions strikes a good balance between sector granularity and the data required to adequately sample and explore each sector. The output of the MLP following HAN is the mean action within each cell. Using a predetermined variance that decays over time, we sample a Gaussian to get $\mathbf{a}_{\text{HA}}^{\text{i}}$; this standard formulation comes from [23].

The key innovation of our model lies in how we combine these individual components. The path tracking embedding, the crowd embedding, $\mathbf{a}_{\text{MPC}}$, and $\mathbf{a}_{\text{HA}}$ are combined to output $\boldsymbol{\alpha}_{\text{raw}} = \begin{bmatrix}\alpha_{\text{raw}}^1 \dots \alpha_{\text{raw}}^6\end{bmatrix}$, where each $\alpha_{\text{raw}}^i \in [0, 1]$, and a categorical distribution $\mathbf{p} = \begin{bmatrix}p^1 \dots p^6\end{bmatrix}$. Note, as in Soft Actor-Critic (SAC), the model learns the mean and log-standard deviation to generate α_{raw}^i, which is then passed through a $\tanh$ function to squash it, followed by scaling and shifting [30]. We then compute $\alpha^i = \text{clip}(\alpha_{\text{raw}}^i + \tanh(\beta), 0, 1)$, where β is a learned parameter. Each $\mathbf{a}_{\text{HA}}^{\text{i}}$ corresponds to α^i and p^i of the same index. After sampling the index j from $\mathbf{p}$, the final action is constructed as a weighted sum: $\mathbf{a} = \alpha^j \mathbf{a}_{\text{HA}}^{\text{j}} + (1 - \alpha^j)\mathbf{a}_{\text{MPC}}$.

Unlike residual DRL, DR-MPC begins with behavior closely matching that of MPC by directly biasing actions toward the MPC controller. This is achieved by initializing the blending coefficient $\beta = -0.8$, which heavily suppresses the learned residual term $\boldsymbol{\alpha}$. However, setting β too low can result in minimal qualitative improvement, as many model updates are required before β increases toward 0—allowing the learned policy to diverge from pure MPC behavior.

In practice, we observe that the model naturally learns to raise β toward 0 over time, gradually gaining the ability to take non-MPC actions. Initially, however, the model lacks any ability to take human-avoidance actions. To alleviate this issue, we incorporate an OOD state detection module alongside a heuristic policy to guide the robot away from potential collisions during early training. This ensures that by the time β approaches 0, the model has been sufficiently exposed to human interactions to begin learning effective avoidance behaviors. Details on the OOD detection and heuristic policy are provided in the following sections.

The DRL algorithm we employ to train our model is a combinatoin of Discrete Soft Actor-Critic (DSAC) and SAC because we have both the categorical distribution $\mathbf{p}$ and the continuous fusion factors $\boldsymbol{\alpha}$ [12, 29]. For DSAC, we have to employ the variant in [98] which uses double average clipped Q-learning with Q-clip to backpropagate through $\mathbf{p}$ in a stable manner. The entropy formulation from SAC is used to optimize the log-standard deviation that generates α_{raw}^i [29, 30]. It is important to note that Figure 4.1 represents the actor model within our actor-critic framework. The critic model shares the same architecture up to the second MLP layer; however, instead of producing $\boldsymbol{\alpha}_{\text{raw}}$ and $\mathbf{p}$ as outputs, it generates the state-action value.

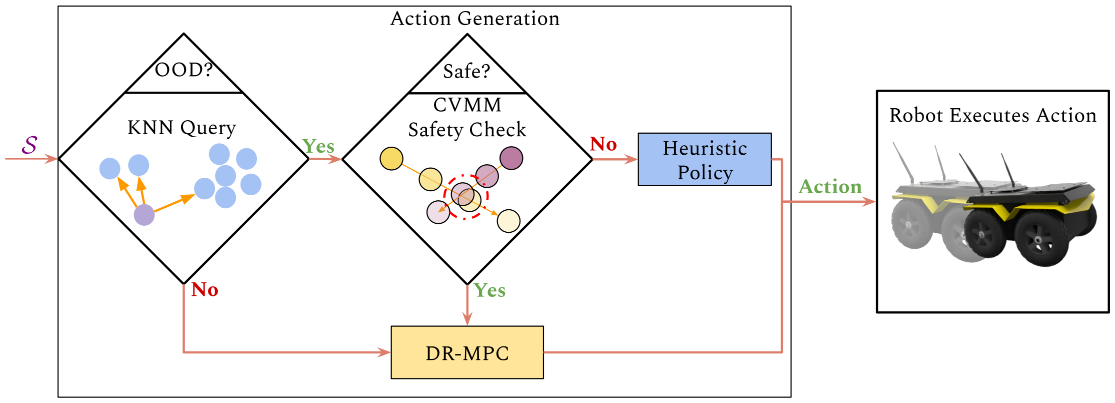

Figure 4.2: DR-MPC Action Generation Block. This action generation block is integrated into the full hardware pipeline, as shown in Figure 3.2. We first determine whether the current state is ID or OOD. If the state is ID, we run DR-MPC directly. If the state is OOD, we apply our CVMM-based safety check to decide whether it is safe to run the DR-MPC action or if we should instead fall back to our heuristic policy.

Additionally, the target action is included as an input to this second MLP.

Although the effective action space of DR-MPC is smaller than regular DRL because of the inductive bias of fusing the learned human avoidance actions and the MPC path-tracking action, this smaller action space contains the best individual actions for path tracking and human avoidance. A fusion of these actions should intuitively perform well on both tasks. While DRL with the entire action space may theoretically achieve higher performance with infinite training data, there are significant practical challenges of tuning exploration versus exploitation to reach true optimality. Thus, we sacrifice exact optimality to significantly reduce learning time while achieving a high-performance policy.

4.4 Model Pipeline

It is important to recognize that the exploration-exploitation problem in DRL does not have to be addressed solely by the model. We can implement additional mechanisms beyond merely sampling from Gaussian distributions in the DR-MPC model. For example, consider the question: which of the six $\mathbf{a}^i_{HA}$ actions should the model should use for fusion? For a randomly initialized model, this choice would be arbitrary, determined by the random weight initialization. Depending on the early ranking of actions dictated by $\mathbf{p}$, the actions chosen during initial exploration, and the critic's learning speed, it could take a long time in the worst-case scenario to discover the most effective human avoidance actions. However, we can design a more intelligent framework to guide the agent toward higher reward regions earlier. The sooner these valuable experiences are stored in the replay buffer, the more effectively the critic can learn the correct Q-values. This concept of early intervention in uncertain or high-risk states has proven effective in interactive imitation learning and sample-efficient DRL [43,60]. This concept is also related to the field of offline-to-online RL where seeding the replay buffer with good demonstrations can lead to faster training [3].

To implement this idea, we perform OOD state detection to assess whether the DRL model has previously encountered a given state. If the state is in-distribution (ID), we allow the DRL model to execute its action. If the state is OOD, we either validate the DRL model's action as 'safe' using a heuristic safety check or override it with an action that is safe according to the heuristic. This process steers the DRL agent toward collision-free areas. Once a state is ID, the DRL agent will continue to explore and may still encounter collisions.

While this pipeline (Figure 4.2) does not guarantee zero collisions, it significantly reduces the likelihood. After experiencing these collisions, the agent should quickly learn to avoid poor reward regions and focus on known good actions, thereby accelerating the learning process. Note that the OOD state detection, heuristic safety check, and heuristic policy are modular components and can be swapped out with other methods or fine-tuned independently of everything else in the pipeline.

4.4.1 OOD State Detection

There are various methods for OOD state detection, such as autoencoders [15], model ensembles [60], and normalizing flows [45]. For our approach, we select the SOTA OOD algorithm: K-nearest neighbors (KNN) in the latent space of our model [76]. We extract the latent embedding from our model at the layer immediately following the fusion of path tracking and human avoidance information. With the Faiss package [17], we can efficiently query the K'th closest vector. A state is considered OOD if its distance to the K'th nearest vector exceeds the threshold: the average distance between ($\mathcal{S}$, $\mathcal{S}'$) pairs in the replay buffer. Intuitively, if a query state is ID, it will roughly have K other states nearby. This method offers two key advantages over approaches like Random Network Distillation (RND) [95]. First, it requires minimal additional compute as it leverages the existing model's latent space without needing to train a new model. Second, by explicitly utilizing all the data, it avoids capacity issues where methods such as RND can 'forget' due to the model's limited capacity.

4.4.2 Heuristic Safety Check and Policy

Upon detecting an OOD state, we assess whether the DRL model's proposed action is safe. We assume a constant velocity motion model (CVMM) for both the human and the robot. We roll out the proposed action for the robot and the estimated human velocity based on the last two timesteps for two seconds. We then evaluate if the safety-human condition is triggered at any point during this rollout; if it is, the action is deemed unsafe. For the safety-corridor condition, we only check the one-timestep rollout result.

If the proposed DRL action is both OOD and fails the CVMM safety check, we evaluate a predefined set of alternative actions for safety (similar to [21]). We select the safe action that is closest to the MPC action, thereby guiding our DRL agent into regions that are both safe and conducive to path advancement.

There are many possible heuristic policies for safety. One alternative we tried is the 'most constraining scenario' heuristic. Given a candidate robot action, we simulate its resulting position and assume that every human agent in the scene attempts to reach that same location. In this manner, we create a conservative estimate: if no human can reach the robot's destination in time to collide, the action is considered safe.

However, during early model development, we empirically found that the CVMM heuristic led to faster learning. One hypothesis for this finding is that ORCA is a cooperative model, so evaluating worst-case human behavior (as in the most constraining scenario) can be overly pessimistic. This conservatism may unnecessarily restrict exploration, thus slowing down the learning process.

4.4.3 Soft Resets

In real-world applications, minimizing downtime is crucial, and resetting the robot to the start of a path after every failure would be inefficient. Therefore, unlike traditional episodic RL, we avoid 'hard resets'. Instead, we return the robot to a non-terminal state either through heuristic rules or manual intervention. The two primary heuristic rules correspond to the safety-corridor and safety-human violations. In the event of a safety-corridor violation, the robot rotates on the spot and reorients itself toward the localization vertex and then slowly moves until it is within the corridor. For a safety-human violation, the robot remains stationary until the dynamic object leaves the safety zone. If the human is uncooperative or unexpected issues arise, a human operator can manually guide the robot back to a safe zone. This approach to training where we start the episode at any non-terminal state is analogous to the Monte Carlo Exploring Starts (MCES) algorithm [77].

4.5 Simulation Experiments

The following simulation experiments quantitatively demonstrate that DR-MPC is more sample efficient than traditional DRL and Residual DRL. These experiments also justify the OOD pipeline.

4.5.1 Simulator and Setup

We used the simulator described in Section 3.2. Our training schema cycles through four scenarios (Figure 4.3). We run our simulator with $n_t = 6$; these 6 humans are modeled using ORCA and continuously move to random goals in the arena. As in [56], because we use a disturbance penalty, the robot is visible to the humans. Our robot has an action space of $v \in [0, 1]$ and $\omega \in [-1, 1]$. The MDP has a timestep of 0.25s, and we train our models with a limited amount of data: 37,500 steps, which equates to around 2.5 hours of driving. We tune and set $K = 100$ for the OOD state detection so that by the end of training, over 95% of the states are ID.

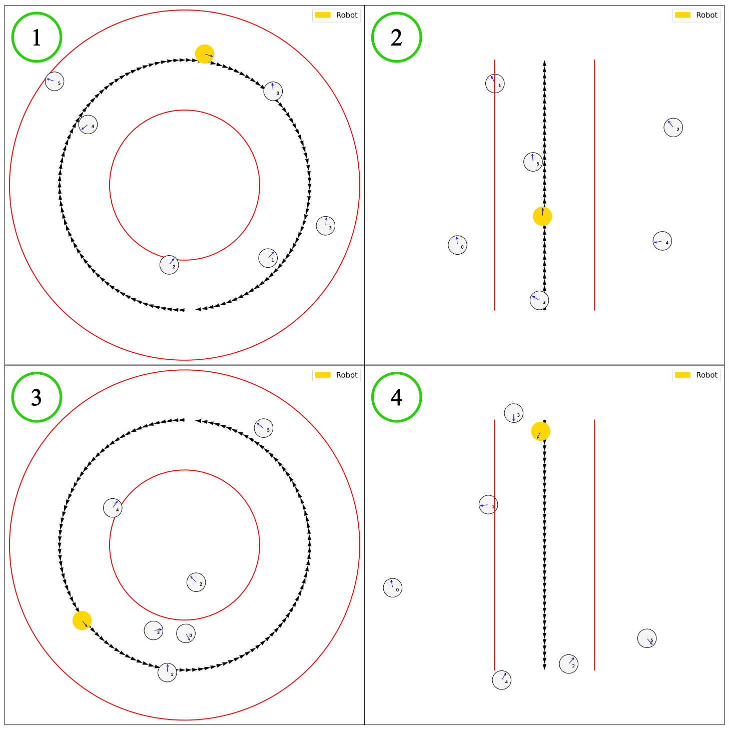

Figure 4.3: Simulation scenarios. The path is black, the corridors are red, the robot is yellow, and the humans are grey. Upon the successful completion of an episode, the system transitions directly to the next scenario without requiring an unnecessary reset or manual repositioning of the robot.

4.5.2 Model Baselines

We evaluate five models. The first is based on [49], which shares a similar backbone with DR-MPC. This model combines the 'PT Embedding' and 'HA Embedding' using an MLP to generate the agent's action. We refer to this model as the DRL model, as it represents the most conventional DRL architecture setup for social robot navigation. The second model is residual DRL [40], which outputs a corrective action: the action executed in the environment is the sum of the MPC and DRL action truncated into the feasible action space. The corrective action has a range of $v \in [-1, 1]$ and $\omega \in [-2, 2]$ to ensure that the combined action can span the entire action space. This choice is motivated by the fact that human avoidance and path tracking are distinct tasks, and we want to guarantee that any necessary action can be represented by the final residual DRL output. The residual DRL model architecture is the same as the DRL model but also has $\mathbf{a}_{\text{MPC}}$ inputted into the final MLP. Then, we evaluate two versions of DR-MPC: one without the OOD state detection and CVMM modules, and one with them. Finally, we compare against ORCA, which, although it does not optimize for the same objectives as our DRL-based models and also has a different action space, serves as an intuitive performance baseline.

4.5.3 Results and Discussion

Figure 4.4 depicts training progress and Table 4.1 is the final model comparison. The plotted lines are the mean values across 10 trials, while the shaded areas indicate the variance observed during training. For all metrics, the values in these plots are averaged over 500 environment steps. We do not use the reward-per-episode metric because different path lengths inherently yield different maximum cumulative rewards, which would skew comparisons. We define the following performance metrics used in our evaluation:

- **SR** (Success Rate): Percentage of episodes in which the robot successfully reaches the end of the path.
- **EDR** (End Deviation Rate): Percentage of episodes where the robot reaches the path's end but is either too far from the goal node or has a significantly misaligned heading.
- **SHRR** (Safety-Human Raise Rate): Percentage of episodes in which the safety mechanism is triggered due to potential human collisions.
- **SCRR** (Safety-Corridor Raise Rate): Percentage of episodes terminated due to the robot approaching the corridor boundary too closely.
- **ATR** (Actuation Termination Rate): Percentage of episodes that end due to the robot freezing or becoming stuck.
- **NNT** (Normalized Navigation Time): Total navigation time divided by the path length, serving as a measure of efficiency.

From the cumulative reward plot, we observe that both DR-MPC models significantly outperform the naïve DRL and Residual DRL models, which have difficulty advancing on the path while avoiding safety-human raises. DR-MPC, however, excels in task switching and optimizes both path tracking and human avoidance using the α parameter. It is worth noting that the DRL and residual DRL models were verified for correctness and, given many more environment steps, could eventually reach similar reward levels as DR-MPC. ORCA guarantees zero collisions with humans modeled as ORCA and achieves the highest success rate. However, ORCA has the highest NNT despite operating in the largest action space: since its action space is (v_x, v_y), the policy can reach more states in a single timestep than the DRL models, which are constrained by the unicycle robot action space. Furthermore, ORCA does not attempt to minimize human comfort (disturbance) or deviation from the path. Deviation is practically useful in the real world where staying closer to the desired path typically aligns better with drivable areas.

As designed, both DR-MPC models begin with a high initial reward due to their initialization with near-MPC path-tracking behavior, which offers better performance than the residual DRL model, which only follows the MPC action in expectation. Nevertheless, the residual DRL model still achieves a significantly higher initial reward than the standard DRL model, due to the guidance provided by its base controller.

Comparing the learning processes of our DR-MPC models, the variant with OOD state detection performs better during the early stages of training, as the heuristic policy guides the robot away from easily avoidable collisions. As training progresses and more states become ID, we observe a temporary rise in safety-human interventions and a slight dip in cumulative reward. However, due to this more intelligent initial data collection strategy, DR-MPC with OOD state detection still maintains a higher cumulative reward throughout training process compared to its counterpart without OOD guidance. Importantly, this demonstrates that the safety mechanism does not compromise long-term performance. Instead, it accelerates learning and minimizes regret by avoiding wasted exploration in irrelevant regions of the state-action space.

One interesting qualitative insight is that, despite the naïve DRL model achieving a high cumulative reward for path advancement, its behavior is often qualitatively inferior to that of the DR-MPC or residual DRL models. Specifically, the DRL policy tends to consistently deviate slightly from the intended path by cutting corners to maximize the path advancement reward at the cost of the deviation penalty. This highlights an additional advantage of the DR-MPC approach—its ease of reward tuning. For instance, when navigating a circular path

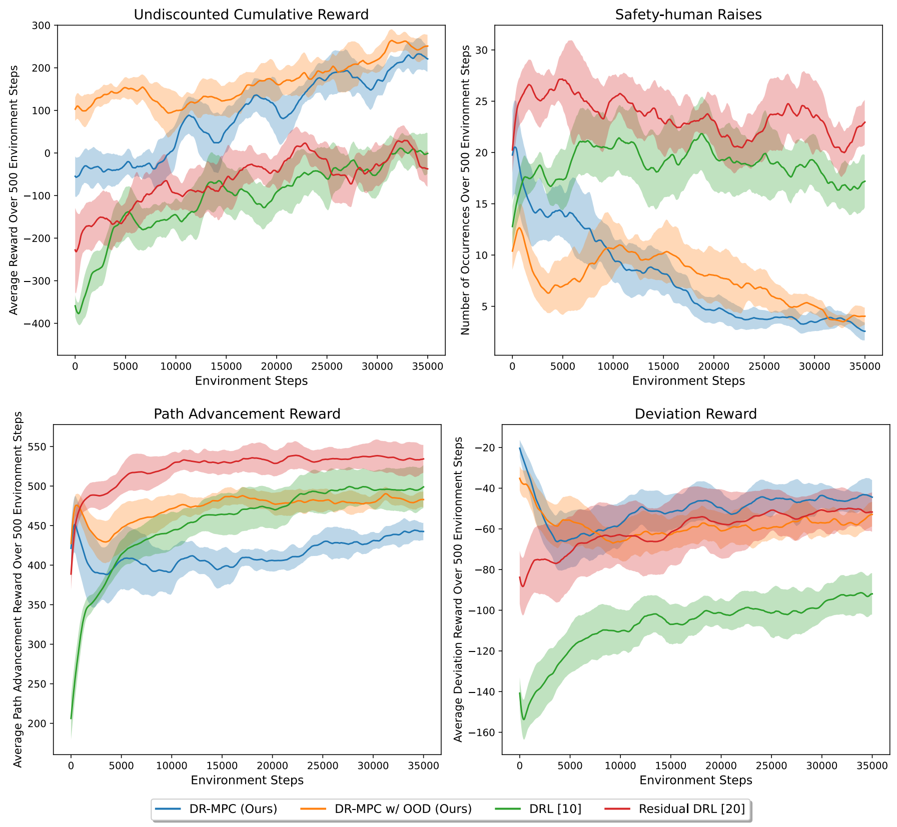

Figure 4.4: Simulation results averaged over 10 trials. Both DR-MPC models outperform naïve DRL and Residual DRL by efficient task switching for human avoidance.

Table 4.1: DR-MPC Simulation Performance Results. Arrows indicate direction of goodness. The asterisk on ORCA's SHRR is because it is guaranteed no collisions.

Model	**SR↑**	**EDR↓**	**SHRR↓**	**SCRR↓**	**ATR↓**	**NNT↓**
ORCA	0.60	0.07	0.00*	0.33	0.00	1.34
DRL [10]	0.20	0.02	0.77	**0.01**	**0.00**	1.16
Residual DRL [20]	0.21	**0.00**	0.79	**0.00**	**0.00**	**1.06**
DR-MPC (Ours)	**0.57**	0.02	**0.30**	**0.00**	0.11	1.32
DR-MPC w/ OOD (Ours)	**0.58**	0.06	**0.31**	**0.01**	0.04	1.17

with no humans present, the DRL model—when optimizing only for path advancement and deviation penalties—learns a policy that tracks slightly inside the circle, as this yields a higher reward under its objective. However, this behavior is not desirable from a design standpoint. In contrast, the DR-MPC model, which incorporates a reference MPC action, adheres more faithfully to the intended path, resulting in qualitatively better trajectories under a wider range of reward parameters.

We qualitatively analyze the OOD state detection module by deploying our trained models from the six dynamic human environment into an environment with two additional static humans placed directly on the path. When the robot approaches a static human, we observe the OOD state detection being triggered, and the heuristic policy guides the robot around the human. In the real-world results (Figure 4.6), the upward trend of the ID (green) line has a spike at 5000 steps, which is not an artifact but reflects early training challenges in latent representation changes making rare state identification difficult. To overcome this, we start with a stricter KNN distance threshold (we scale the threshold calculation by $\frac{1}{3}$) and gradually relax this scaling factor to identity by the end of training. This tuning leads to better results than a constant distance threshold.

Performance could be further improved with a better heuristic safety check and heuristic policy. Currently, the CVMM safety check is still overly conservative: early in training, 21% of CVMM triggers result in DR-MPC collisions on the next timestep, dropping to 5% by the end of training. Similarly, early in training when the heuristic policy (which relies on CVMM) causes a collision, DR-MPC also collides 90% of the time, but this decreases to 3% by the end of training. These findings suggest that CVMM struggles to model close-contact ORCA behavior. However, the CVMM safety check remains valuable in real-world scenarios, particularly for inattentive humans walking straight or standing still.

4.5.4 Reward Ablation

Given our new decision process formulation, we analyze the contribution of each component of the reward function. Table 4.2 compares the full *DR-MPC w/ OOD* model (using all reward terms) to ablated versions where individual rewards are removed.

Interestingly, removing r_{pa} still results in a high SR, likely because the policy begins with

Table 4.2: Reward Ablation Study on DR-MPC with OOD Module

Model	**SR**$\uparrow$	**EDR**$\downarrow$	**SHRR**$\downarrow$	**SCRR**$\downarrow$	**ATR**$\downarrow$	**NNT**$\downarrow$
Base	0.58	0.06	0.31	0.01	0.04	1.17
r_{pa}	0.46	0.02	0.42	0.00	0.09	1.60
r_{dev}	0.46	0.08	0.19	0.06	0.21	1.27
r^*_{goal}	0.56	0.07	0.37	0.01	0.00	1.17
$r^*_{\text{cor-col}}$	0.56	0.06	0.27	0.05	0.05	1.24
r^*_{act}	0.56	0.09	0.24	0.04	0.08	1.30
$r^*_{\text{hum-col}}$	0.24	0.00	0.76	0.00	0.00	1.07
r_{dist}	0.59	0.13	0.21	0.00	0.07	1.54

MPC path-tracking behavior, allowing it to reliably reach the end of the path. However, since we do not include a goal-specific reward, removing r_{pa} significantly increases the NNT, as expected.

When the goal penalty is removed, EDR increases slightly. This is intuitive: without being penalized for reaching the goal incorrectly, the model completes more episodes—but the increase is marginal, likely because MPC-based path tracking already leads to proper episode completion.

Eliminating r_{dev} increases both EDR and SCRR, as expected, since the agent is no longer penalized for deviating off the path, allowing it to maneuver closer to the corridors. Removing $r^*_{\text{cor-col}}$ leads to a higher SCRR, again consistent with expectations.

Removing r^*_{act} leads to a higher ATR, along with increased NNT and decreased SHRR. This outcome occurs because the robot adopts a more conservative behavior, often waiting longer for humans to pass, which results in slower progress but fewer human collisions. While this suggests that omitting r^*_{act} may offer some benefits, further testing is needed to determine whether the algorithm can consistently converge to a reliable policy and avoid the freezing robot problem.

Unsurprisingly, removing $r^*_{\text{hum-col}}$ causes a drastic increase in SHRR, highlighting its critical role in collision avoidance. Finally, removing r_{dist} reduces SHRR and greatly increases both SR and EDR. This makes intuitive sense: r_{dist} primarily encourages comfort by discouraging proximity to humans, rather than directly affecting success rates. Although it quantitatively appears to hurt performance, its qualitative influence on human-aware behavior remains important.

Overall, we conclude that each reward term plays an essential role in enabling robust,

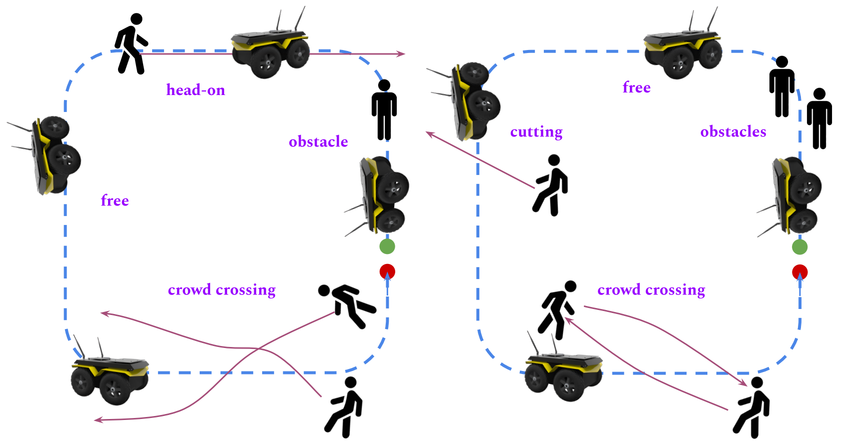

Figure 4.5: Real-world testing scenarios used to evaluate models. These loops contain diverse situations for social navigation.

socially-aware navigation, and that their combined effect is necessary for real-world deployment.

4.6 Hardware Experiments

4.6.1 Experimental Setup

Our hardware setup is detailed in Section 3.3. During training, between zero and four humans interact with the robot at any given time. These interactions include behaviors such as standing in the robot's path, walking toward it, or walking alongside it. In total, approximately eight different individuals participated in the training process, each exhibiting their own varied style of movement and interaction.

The MDP timestep is 0.2s, aligned with incoming LiDAR data every 0.1s. Although we trained on 3.75 hours of experience transitions, the entire process took about 15 hours. After every 500 experience tuples, we paused data collection to allow the model to finish updating on the existing data, performing twice as many training updates as environment samples. Additionally, outliers caused by processing delays from resource allocation variance were filtered to reduce data noise. Also, soft resets, along with charging the power bank and robot batteries, contributed to the additional time.

We periodically evaluate the model on a set of fixed yet diverse scenarios (Figure 4.5). In the scenario on the left, the robot must first navigate around a stationary human, then evade an aggressive human walking directly toward it, continue along the path in a free-driving segment, and finally pass through a crossing crowd. The scenario on the right assesses the robot's ability to weave between two stationary humans, followed by another free-driving segment, a single pedestrian crossing its path, and a differently configured crowd crossing. Each model is evaluated over six runs—three per scenario.

We benchmark real-world DR-MPC against three models: (1) DR-MPC trained in simulation without modification ('Sim Model Raw'), (2) DR-MPC trained in simulation with real-world adjustments, such as observation delays, acceleration constraints, and distance-based detection limits ('Sim Model Adjusted'), and (3) a heuristic policy, which executes $\mathbf{a}_{\text{MPC}}$ if it passes the CVMM safety check and switches to the heuristic policy used to guide the DR-MPC model if unsafe.

The real world introduces challenges absent in simulation, such as perception errors (missed and false detections), variable processing delays (localization, human detection, action generation), and probabilistic human motion. These challenges highlight a key advantage of DRL: by maximizing expected cumulative reward, DRL can effectively learn despite the noise inherent in the collected (s, a, s', r) tuples.

Table 4.3: Real-world Policy Results ($\mu \pm \sigma$ # Per Loop)

Algorithm	**Steps to Goal** $\downarrow$	**Safety-human Raises** $\downarrow$	**Safety-corridor Raises** $\downarrow$
Heuristic	158.0 ± 2.7	1.0 ± 0.6	0.3 ± 0.5
Sim Model Raw	175.3 ± 3.4	0.5 ± 0.5	3.2 ± 1.1
Sim Model Adjusted	161.2 ± 4.7	0.7 ± 0.5	1.5 ± 0.8
DR-MPC (Ours)	**146.3 ± 3.1**	**0.3 ± 0.5**	**0.0 ± 0.0**

4.6.2 Results and Discussion

Table 4.3 shows DR-MPC outperforming benchmark models in all key metrics—fewer safety raises and faster navigation. The training process (Figure 4.6) begins with performance nearly identical to the heuristic policy, which is as expected because every state is OOD in the beginning, so the heuristic policy is always active. During exploration, DR-MPC performs worse but eventually coalesces its experience to achieve superior results. We notice β (orange) trends toward 0, while the average α (blue) remains low, indicating the model learns when to best select the MPC action. After 4 hours of data, DR-MPC's performance begins to plateau. While more data would likely add marginal improvements, social navigation is a long-tail problem, and even in simulation with deterministic humans, safety-human raises never reach zero.

Qualitatively, DR-MPC exhibits key crowd-navigation behaviors: smoothly weaving around still humans (Figure 4.7 left and middle) and deviating from its path and slowing down for moving humans (Figure 1.1 and Figure 4.7 right). Also as desired, in human-free areas, DR-MPC switches to MPC path tracking.

The performance of the heuristic policy degrades due to noisy human velocity estimates, which stem from inferring position based on the bounding box center. This center shifts depending on body orientation. For example, imagine standing in front of the robot and rotating in place. When you face the robot head-on, your chest is the closest point. However, when you turn sideways, your shoulder or arm becomes the closest point, causing the bounding box center—and thus the inferred position—to shift closer to the robot, even though the center of your body has not actually moved. Also, this policy does not account for delays or the robot's acceleration, reducing its ability to navigate around humans. However, it still guides DR-MPC towards open spaces, which results in overall faster learning.

'Sim Model Raw' performs poorly, oversteering and colliding with virtual boundaries due to unaccounted state delays. Adjusting the simulator ('Sim Model Adjusted') improves performance but still falls short of DR-MPC trained in the real world. This underscores the advantage of real-world training in handling noisy data and developing robust policies.

4.7 Limitations and Future Work

One limitation is the manual tuning of the heuristically defined reward function, particularly balancing path-tracking and human-avoidance rewards. For instance, overemphasizing collision penalties impedes progress along the path; Imitation Learning methods bypass this limitation by replicating expert behaviour [41]. Future work includes incorporating reward learning methods, such as Inverse RL [46] and preference learning [89], to enable DR-MPC to acquire

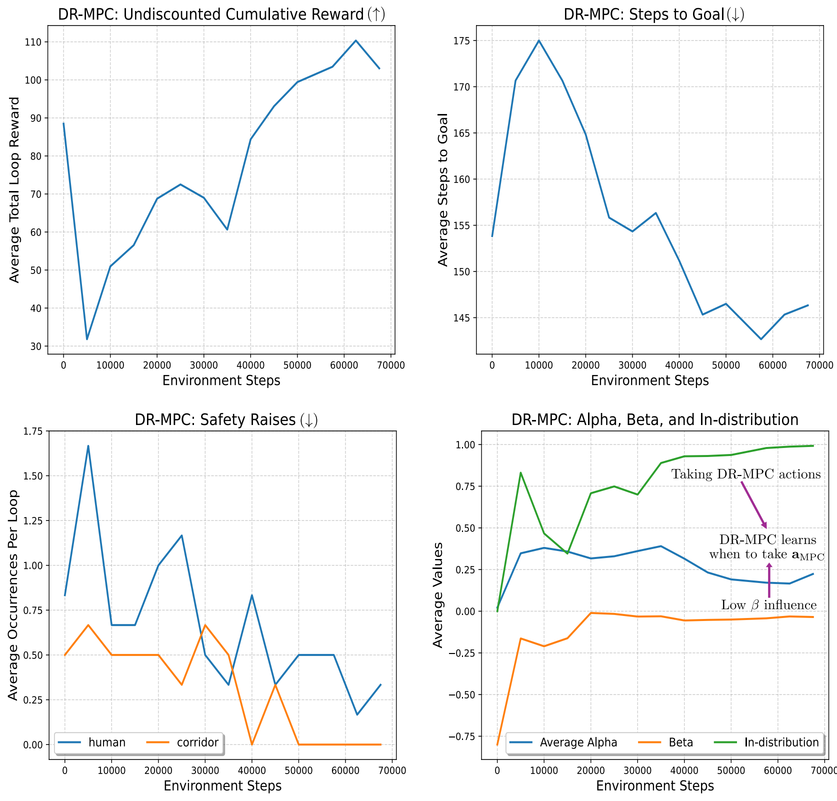

Figure 4.6: DR-MPC's real-world training results averaged over 6 trials—3 per testing scenario. DR-MPC starts with performance equal to the heuristic policy. However, because of DRL's ability to learn through noise, after exploration, the final model performs better on each key metric compared to where it started.

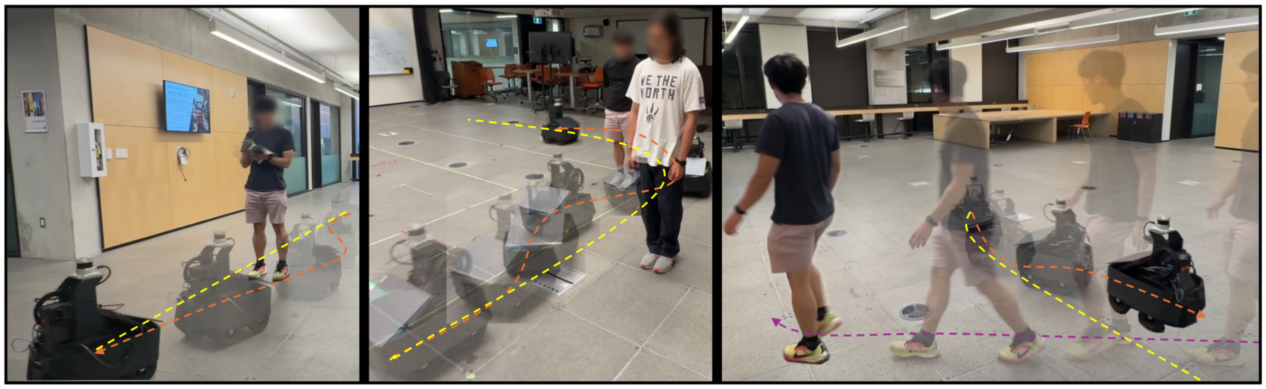

Figure 4.7: Real-world examples. Reference path is yellow, robot's trajectory is orange, and human's trajectory is purple. Left: the robot smoothly navigates around a human. Middle: robot navigates around two static humans then returns to its path. Right: robot deviates from its path to avoid disturbing the human.

more human-aligned behaviors.

Another limitation involves false collision detections due to modeling humans as circles. False positives occur because humans are often wider shoulder-to-shoulder than front-to-back. False negatives arise when extended feet during walking or resting postures cause the center point to misrepresent their state. Future work includes skeleton detection to better capture nuances, improving collision accuracy.

Lastly, we acknowledge that our ‘real-world’ experiments capture only a subset of human behaviors expected in true ‘in-the-wild’ scenarios. Our results reflect interactions with people familiar with the robot. Behaviors such as stopping to observe the robot or intentionally obstructing its path are not encompassed in our current environment. In future work, we aim to conduct ‘in-the-wild’ experiments to evaluate the robot’s performance with more diverse interactions.

4.8 Summary

In this chapter, we make the following key contributions:

1. **Deep Residual Model Predictive Control (DR-MPC):** a novel integration of MPC path tracking with DRL that preserves the ability to learn human avoidance behaviors while dramatically accelerating DRL training for social robot navigation.
2. **Outer-loop OOD state detection and heuristic policy:** a mechanism that reduces collisions during training without compromising final performance.
3. **Real-world DRL training for social navigation:** we demonstrate DR-MPC’s effectiveness in both simulation and hardware experiments, and to the best of our knowledge, this is the first work to train a DRL-based social navigation agent directly in the real world, thereby bypassing the sim-to-real gap and addressing the mismatch between modeled and actual human dynamics.

These contributions advance the field by showing that while simulation remains an indispensable tool for model development, DRL agents can be trained and deployed to operate effectively in real-world environments, where the most critical challenges ultimately lie.

Chapter 5

The Ingredients Needed for Social Robot Navigation with Imitation Learning

5.1 Introduction

Imitation Learning (IL) has seen remarkable success, from early work on lane following in 1988 [65] to recent advances in dexterous manipulation with vision-language-action models (VLAs) in 2025 [38]. While IL has proven effective in many domains, there remains significant ambiguity about why simple methods such as Behavioural Cloning (BC) succeed in some settings but fail dramatically in others [14, 91]. In our experience, vanilla BC—implemented by minimizing a MSE loss (Equation 2.3)—yields poor policies. To make BC effective for social robot navigation, we find that several key design choices and implementation details are essential.

We focus on IL-based social robot navigation because designing reward functions that encode social norms remains a difficult and largely unsolved problem. If IL methods can succeed, they offer an attractive alternative: bypassing the challenge of hand-crafting complex reward functions by simply learning from expert demonstrations.

Another significant benefit of using IL is the ability to safely collect data in complex scenarios, even in challenging in-the-wild environments. In contrast, DRL poses substantial safety concerns when deploying untrained models in real-world settings, as these models must explore and learn from interactions.

5.2 Related Works

In this section, we motivate several design choices for successful IL in social robot navigation.

5.2.1 Avoiding Simple Policies for Imitation Learning

A recent analysis of BC with continuous action regression found that training "simple" policies can perform poorly in long-horizon tasks due to compounding error [73]. The authors propose three effective alternatives. First, they suggest using a discrete action space, which typically results in lower error accumulation over time. Next, they demonstrate that diffusion-based policies produce more robust policies than conventional regression. Third, they show that having large state coverage greatly aids performance.

As a result, we design an action space that contains both discrete and continuous actions. In addition, we use data augmentation that does not compromise data quality, which aids in increasing the state coverage.

5.2.2 Action Chunking

Action chunking refers to predicting a sequence of future actions instead of just the next immediate action. Several works have shown the benefits of this strategy in IL [11, 44]. [11] explores the trade-offs of varying the prediction horizon in diffusion-based policies and finds an "inverted-U" relationship between horizon length and policy performance: very short or very long horizons perform worse than intermediate ones. Similarly, in [44], action chunking improves fine-tuning of VLAs by encouraging better temporal coherence in policy output. Thus, we integrate and evaluate action chunking.

5.2.3 Plasticity in Neural Network Training

An emerging challenge in continual learning is the "loss of plasticity": the network's diminishing ability to learn as training progresses [16]. In their study, [16] trains a model on a sequence of binary classification tasks, where the target class pair changes over time. They observe that a network trained on prior tasks performs worse on the current task than one trained from scratch. This result is surprising, as prior training could intuitively serve as a form of pre-training. The finding suggests that the model ‘forgets’ how to learn. To address this, they propose using high weight decay and introducing random stochastic perturbations to the model parameters—techniques that help preserve model plasticity.

A similar insight can be found online RL: in [70], the policy network is periodically perturbed toward a randomly initialized network. While part of the performance gain may come from increased exploration when the model weights are perturbed, it is plausible that some benefit derives from counteracting the loss of plasticity identified in [16].

Inspired by these findings, we adopt a similar regularization strategy: periodically nudging our network weights toward a randomly initialized network. We find this both improves generalization and significantly enhances closed-loop performance when training with a real-world dataset. We also alter this design slightly with some inspiration from evolutionary algorithms. From the results, we provide a hypothesis for why decaying plasticity affects our stationary supervised learning problem.

5.2.4 Data Augmentation

ML models are known to scale effectively with dataset size, and data augmentation is a widely used technique to expand an existing dataset without additional data labeling [90]. Since BC is a form of supervised learning, its performance similarly benefits from larger datasets. However, data augmentation in IL poses a unique challenge: modifying the state often necessitates recomputing the expert action, unlike in standard classification tasks where the label typically remains unchanged. Since collecting expert actions can be a significant bottleneck in IL, this makes effective augmentation more difficult.

One successful example of data augmentation in IL comes from [26], where demonstrations are recorded simultaneously from three cameras—left, center, and right. The expert actions associated with the side cameras are adjusted to correspond to what the robot would do if it were in that shifted viewpoint, effectively tripling the data while preserving moderate action quality.

We later explain our flip augmentation procedure that doubles the dataset without degrading the quality of the expert labels, making it a highly cost-effective augmentation strategy for human avoidance and boosting generalization to tracking different paths.

5.2.5 Imbalanced Datasets

We can view our task as a multi-class decision process, where each state maps to one of three high-level behaviors: pure path tracking, pure human avoidance, or a combination of both. In our later section on data collection, we break down this imbalance for our dataset.

This class imbalance leads the model to rely more heavily on path-related features in the state, often ignoring the human-related components. Drawing from the insights of [33], models often prioritize feature ‘availability’ over ‘predictivity’; one notion of availability tested in [33] is the size of the feature vector. That is, larger features can dominate the learned representation, even if they are not the most informative for the task.

5.3 Methodology

The core of our model architecture comes from [32] (Figure 4.1), which elegantly handles both the varying number of humans and their variable history lengths. The architecture is identical up to and including the first MLP of the fusion stage; however, the output of this MLP now directly parameterizes the actions.

5.3.1 Combining Discrete and Continuous Actions Spaces for Efficient Action Chunking

As highlighted in [16], BC tends to yield lower compounding errors when applied to discrete action spaces compared to continuous ones. However, discrete actions introduce a trade-off: they partition the action space into coarse bins, which can limit the granularity of control. Increasing the number of bins can mitigate this by refining the resolution, but this in turn increases the output dimensionality, making the learning problem more complex and data-hungry.

Another key drawback of discrete action spaces lies in the inefficiency of action chunking. For an action horizon of H and n discrete bins per timestep, the total number of possible action sequences grows exponentially as n^H. For example, with only 6 bins per timestep—comprising 2 linear velocity options and 3 angular velocity options—over a 5-step horizon, we get $6^5 = 7776$ unique action sequences. Despite collecting 400,000 environment steps (detailed in the following data collection section), such a large discrete space is sparsely covered, with many sequences likely receiving few or no samples. This renders purely discrete approaches statistically inefficient.

To address this, we propose a hybrid approach that combines discrete and continuous action representations similar to the HAN in DR-MPC [32]. In our setup, each discrete bin corresponds to a specific region in the action space, within which a continuous action is learned. This method retains the benefits of discrete modes, while allowing for fine-grained control within each mode.

In the context of social robot navigation, multi-modal behaviors often manifest in the angular velocity: a robot may choose to turn left, turn right, or remain stationary while waiting for a human to pass. Based on this intuition, we discretize angular velocity into three equal bins per timestep: left, straight, and right. Linear velocity remains continuous in the range $[0, 1]$, while angular velocity is partitioned into:

- Left: $\left[v_{\text{min}},\ v_{\text{min}} + \frac{v_{\text{max}} - v_{\text{min}}}{3}\right]$
- Straight: $\left[v_{\text{min}} + \frac{v_{\text{max}} - v_{\text{min}}}{3},\ v_{\text{max}} - \frac{v_{\text{max}} - v_{\text{min}}}{3}\right]$
- Right: $\left[v_{\text{max}} - \frac{v_{\text{max}} - v_{\text{min}}}{3},\ v_{\text{max}}\right]$

With three discrete bins per timestep and an action horizon of $H = 5$, we now have only $3^5 = 243$ discrete path options—an order of magnitude smaller than the previous 7776, enabling more tractable learning and planning.

Formally, our model outputs a vector of dimension $3^H \times H \times 2 + 3^H$. We reshape this output into two components: an action tensor $\mathbf{A} \in \mathbb{R}^{3^H \times H \times 2}$, where each row encodes a sequence of continuous linear and angular velocities along a candidate path and a probability vector $\mathbf{p} \in \mathbb{R}^{3^H}$, which represents the model's confidence in each discrete path. Each row in $\mathbf{A}$ is computed such that the angular component is properly squashed, scaled, and shifted to lie within the corresponding angular velocity bin. Note that this process can be and is fully vectorized in our implementation.

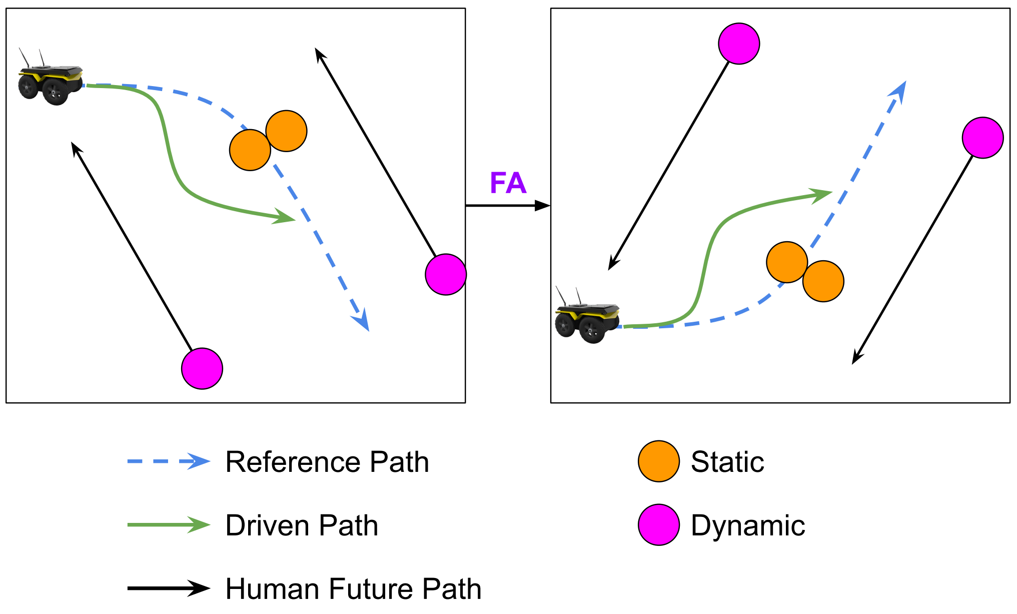

Figure 5.1: Flip Augmentation Example. A visual demonstration of the flip augmentation procedure. Both scenarios are in the robot's frame. The humans, path, and expert's driven path are flipped about the robot's y-axis.

We define the loss function as:

$$L_{\text{combined}} = CE(\mathbf{p}, i) + \sum_{j=1}^{3^H} p_j \times MSE(\mathbf{A}[j, :, :],\ \mathbf{a}^*) \tag{5.1}$$

Here, $CE(\mathbf{p}, i)$ is the cross-entropy loss for the ground truth path index i, encouraging the model to assign high probability to the correct discrete path. The second term is a weighted mean squared error (MSE), where the predicted continuous actions $\mathbf{A}[j]$ for each path is compared against the ground-truth continuous action sequence $\mathbf{a}^* \in \mathbb{R}^{H\times 2}$. This loss is weighted by the model's confidence p_j in that path. This encourages the model to make accurate predictions along paths it believes are most likely.

This hybrid representation strikes a balance between discrete and continuous modeling, offering the multi-modality and reduced error compounding of discrete spaces with the control precision of continuous actions.

5.3.2 Data Augmentation Through Flips

Our state representation is rotationally invariant: past human trajectories, the robot's trajectory, and the planned path are all expressed in the robot's current frame. A simple yet effective data augmentation technique is to horizontally flip the state about the robot. Specifically, we negate the y-coordinates of all components (robot, humans, path) and invert the sign of the expert action's angular velocity (Figure 5.2. This transformation preserves the validity of the state-action pairs while doubling the dataset.

This flip augmentation introduces two key benefits. First, it enhances path diversity—for example, if the original dataset contains only CW paths, the augmentation introduces CCW variants. Second, it increases the coverage of the expert's state-action distribution. For instance, if the original data shows the expert passing to the left of a static human, the augmented data introduces a mirrored scenario where the robot passes on the right.

Another possible augmentation strategy is to synthetically insert additional humans into the scene, provided they do not affect the robot's optimal action—for example, by placing a human far from the robot's path. While this increases the size of the dataset, we hypothesize that it would not significantly improve the model's final performance. Learning to ignore distant humans is relatively trivial, so such additions may offer limited benefit. A more promising direction could be the use of generative models to create realistic, diverse human-robot interaction scenes. However, a key limitation remains: we still require a human to annotate the expert action for each generated scenario. This is a difficult process, as it is often easier to teleoperate the robot in real time than to observe a static frame and determine the correct expert action—humans are intuitive drivers, but not necessarily good post hoc annotators. We leave the exploration of other data augmentation strategies to future work.

5.3.3 Traditional Machine Learning Parameters

We find that our network and dataset require a small learning rate of 10^{-4} and approximately two million training steps. Attempts to reduce the number of gradient steps by increasing the learning rate result in highly unstable training performance.

As motivated by [16, 70], we opt for a larger network with increased weight decay to encourage generalization. Additionally, we find that using the AdamW optimizer is necessary for proper decoupled weight decay regularization, as standard Adam does not correctly apply weight decay [52].

5.3.4 Injecting Randomness Into the Model

Every 50,000 training steps, we instantiate the network using PyTorch's orthogonal initialization and move the current model's weights towards the random network by 30% [62]. Hence, $w_{\text{model}} \leftarrow 0.7w_{\text{model}} + 0.3w_{\text{random\_network}}$.

An additional ingredient that we apply to our weight perturbation scheme is resetting the current model to the best-performing checkpoint (based on a metric such as closed-loop success rate or validation loss), and then applying the perturbation to its weights. This approach is inspired by techniques in evolutionary algorithms, where mutations are applied to the most fit individuals to promote exploration while preserving strong baseline performance. Over time, we gradually decay the size of these perturbations.

5.3.5 Reducing the Path-Tracking Latent Dimension

In practice, the real-world dataset we collect is imbalanced, with 73% of the demonstrations reflecting the robot simply following T&R's MPC controller. This imbalance implies that a BC model that only attends to the path embedding could still achieve a relatively low loss, without meaningfully learning to account for human interactions.

We also observed this phenomenon empirically: when grouping similar states based on the latent embeddings of the first layer after fusing the human-avoidance and path-tracking information from our trained model, we found that states were more often clustered by path structure than by human configuration. In other words, states that are close in the latent space tend to share similar path geometries, rather than similar human-robot interactions. While this does not necessarily prevent the BC agent from predicting good actions, it suggests a potential over-reliance on path information.

To mitigate this potential issue, we apply a simple architectural intervention inspired by [33]: we reduce the dimensionality of the path-tracking embedding. This limits the represen-

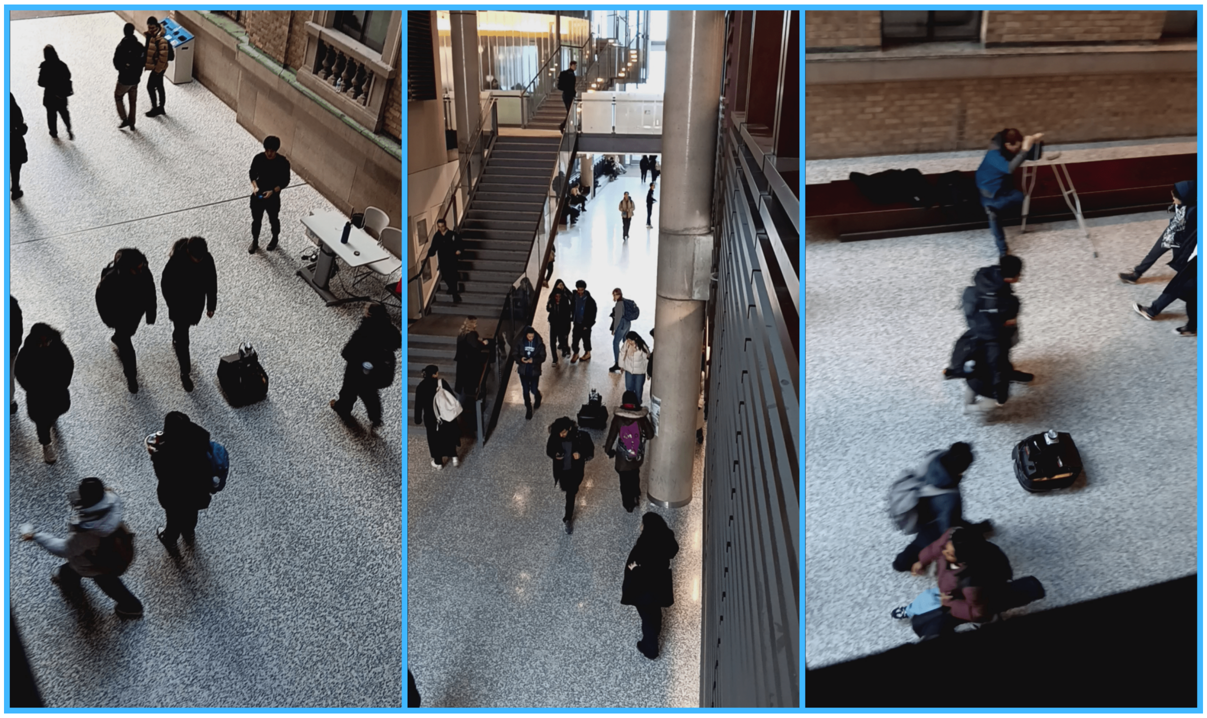

Figure 5.2: IL Data Collection Examples. Examples of data collection on the university campus between class times. The constrained spaces with respect to the number of humans caused many robot-human interactions.

tational capacity available for encoding path information, encouraging the model to balance its attention more evenly between path and human features during learning and inference.

5.4 Hardware Experiments

5.4.1 Data Collection

Our hardware setup is described in Section 3.3. The human teleoperator controls the robot at a frequency of 10 Hz, matching the rate of incoming LiDAR packets and, consequently, the control frequency of the IL agent.

To streamline data collection, we leverage T&R. By default, the robot autonomously follows the reference path using T&R, while a human operator can seamlessly override its actions when necessary. A well-tuned MPC path tracker generally outperforms a human in pure path-following tasks. This is because the teleoperator does not have physical access to the full reference path, making precise tracking more difficult. As a result, the operator primarily intervenes to manage social interactions, such as navigating around pedestrians. Once the robot has safely passed nearby humans, control is smoothly returned to the MPC path tracking, which resumes accurate path following.

We collected just over 11 hours of data over two weeks on the university campus. To maximize the frequency and diversity of human-robot interactions, we prioritized data collection during peak pedestrian traffic—such as the transition periods between classes. These times typically range between 5 to 30 people around the robot. We also collected lower density scenes of 0 to 5 people. To further enhance diversity, we also collect data both indoors and outdoors on the campus.

This in-the-wild data collection resulted in significantly more diverse interactions than those observed in prior works such as DR-MPC. For example, we encountered pedestrians intentionally obstructing the robot's path, groups walking alongside the robot, clusters of people engaged in conversation, and individuals waving at the sensor, crouching to tie their shoes, or appearing distracted while using their phones. These real-world scenarios capture a richer spectrum of human behavior and social context, providing a more challenging and representative dataset for learning.

5.4.2 Evaluation

To quantitatively evaluate our method for closed-loop evaluation, we use simulation. Our evaluation suite consists of 27 unique scenarios, constructed by combining three different path types with varying numbers and types of humans in the environment. The three paths include a straight 8-meter path, a CW circular path with an 8-meter diameter, and a CCW circular path of the same size.

For each path, we test the robot's behavior under different levels of human presence. Specifically, we vary the number of static humans directly on the path between one, two, and three. Separately, we introduce either zero, three, or six dynamic humans moving through the scene. For these dynamic pedestrians, we maintain a 2:1 ratio of "regular" to "aggressive" behavior profiles. Aggressive humans are intended to simulate distracted individuals—such as those using their phones—or those who intentionally challenge the robot. To model this, we reduce their ORCA safety margin to just 5 cm. The idea is that a human not paying attention may only notice the robot when it is directly in their personal space.

Although the evaluation is conducted in simulation, we acknowledge that this introduces a domain gap—similar to the sim-to-real problem, but in the reverse direction. The simulated humans follow different movement patterns compared to those observed in the real world; for example, they navigate between random destinations around the circle. Moreover, the simulation lacks realistic sensor noise and processing delays. While these differences might seem advantageous, from a ML perspective, it may be harmful. These differences introduce distributional mismatch between the training and evaluation conditions, which may affect performance in unpredictable ways.

Despite these limitations, simulation allows us to evaluate our models under consistent and repeatable conditions across a wide range of scenarios. Conducting such evaluations in the real world would require significant effort and coordination, particularly when comparing across multiple model ablations. Additionally, it would be nearly impossible to reproduce the same human behavior for each trial—humans would inevitably respond differently when interacting with the first versus the last model tested. For these reasons, simulation provides a practical and scalable alternative for holistic evaluation.

We evaluate a total of 11 models. We begin with a 'Base' model, which corresponds to standard BC that performs regression on the one-timestep expert action using a MSE loss.

To determine the effect of combined discrete and continuous action space, we evaluate the following three models:

- The 'Combined' model integrates all key design components and employs the hybrid action space consisting of discrete angular bins and continuous actions constrained within those bins. We use 3 discrete bins and an action horizon of 3, resulting in a total of $3^3 = 27$ candidate paths.

- The 'Discrete' model uses a larger discrete action space of 25 bins with a shorter action horizon of one to maintain tractability. We keep all other ingredients active.

- The 'Continuous' model relies solely on a continuous action space with an action horizon of three, and likewise retains all other components.

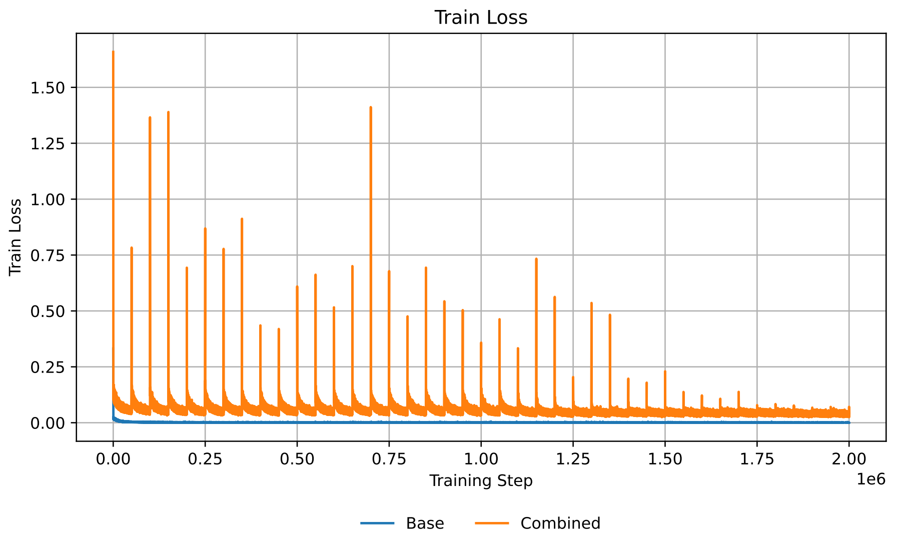

Figure 5.3: BC Training Loss. The training loss for the 'Combined' model plotted is the model's continuous action loss. The large spikes correspond to periodic weight resets used as part of the training strategy.

To assess the contribution of each component, we perform ablations from the 'Combined' model, removing one element at a time. For the weight reset mechanism, we begin with a 30% perturbation toward randomly initialized weights and apply this every 50k training steps. After 750k training steps, we begin to anneal this perturbation percentage by a factor of 0.9 every 50k steps.

The HA embedding is size 128, and the PT embedding is size 16. In the smaller PT embedding ablation, we increase the PT embedding size to 128. For the AdamW ablation, we replace the optimizer with Adam. The weight decay ablation sets the weight decay to zero instead of the default value of 5×10^{-3}. Lastly, in the action chunking ablation, the action horizon is reduced from three to one.

5.4.3 Real-to-sim Results

The performance gap between the 'Base' model and the fully equipped 'Combined' model is substantial: the 'Base' model achieves a success rate of 27.8%, while the 'Combined' model reaches 96.3%. The training losses for both models are shown in Figure 5.3. The loss plotted for the 'Combined' model corresponds to its continuous action prediction over an action horizon of three, whereas the 'Base' model predicts a single timestep. The large spikes observed in

Table 5.1: BC Real-to-sim Results. Comparing the 'Base' model (naïve BC) to our full model including ablations over action space and other key architectural choices.

Model	**Success Rate (↑)**
Base	27.8
Combined	**96.3**
Discrete	91.7
Continuous	93.5
– Reset	59.3
– Flip Data Augmentation	78.7
– Action Chunking	93.5
– Small PT Embedding	**96.3**
– Weight Decay	92.6
– AdamW	26.9
– Use Best at Reset	**96.3**

the 'Combined' model's loss curve result from periodic weight resets during training. As the reset perturbation is annealed over time, these spikes diminish and eventually disappear.

From Table 5.1, we observe that the choice of action space—discrete, continuous, or combined—has only a modest impact on performance. This suggests that the standard purely continuous action space is viable despite the multi-modal training data, which is a surprising finding.

From the ablation study, we find that resetting model weights, applying data augmentation, and using AdamW are critical design choices. In particular, we attribute the importance of AdamW to its role in proper weight decay regularization [52]. The success rate of these models over training is presented in Figure 5.4. From the figure, we observe that models trained without AdamW or without weight resets exhibit stagnation, with little improvement in success rate beyond the initial phase of training.

Although BC is framed as a standard supervised learning problem, we interestingly find that periodically injecting variability into the model (perturbing weights toward a random network) significantly improves performance. We hypothesize that this benefit arises from the inherently multi-objective nature of social robot navigation, which requires the policy to simultaneously learn human avoidance and path tracking. This is in contrast to traditional classification tasks—where the learning objective remains consistent across the dataset.

Injecting variability into the model helps maintain learning capacity across both objectives.

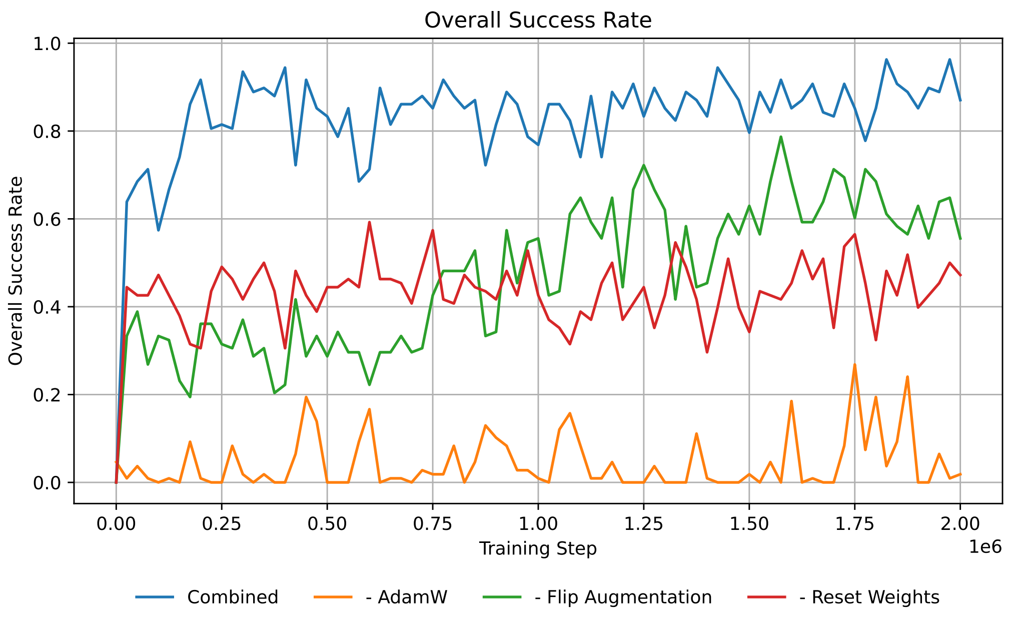

Figure 5.4: BC Real-to-sim Ablation of Key Ingredients. Raw success rate of key design choices throughout training. The results show that each ingredient contributes meaningfully to overall performance, with all components being essential to achieving the high success rate observed in the ‘Combined’ model.

This aligns with findings from [16], which show that networks lose plasticity when repeatedly trained on changing targets. If we view BC in social navigation as a form of continual or multi-task learning—where the task emphasis shifts based on the sampled batch—then this form of regularization may counteract the degradation in performance associated with plasticity loss. To fully validate this hypothesis, we would need to either identify a similar multi-objective task and perform the weight reset ablation, or use our current setup with a batch size equal to the dataset size to eliminate changes in task distribution across minibatches. The latter would be computationally intensive, as it requires computing gradients over the entire dataset at each step and would likely involve loading and unloading data to fit within GPU memory constraints. We leave these analyses for future work.

Moving on, we observe that action chunking and weight decay appear to have only a minimal effect on overall success rate.

We also find that both using a smaller path-tracking embedding and resetting to the best-performing weights during training independently achieve success rates comparable to the full ‘Combined’ model. This suggests that these design elements may not be as critical as initially

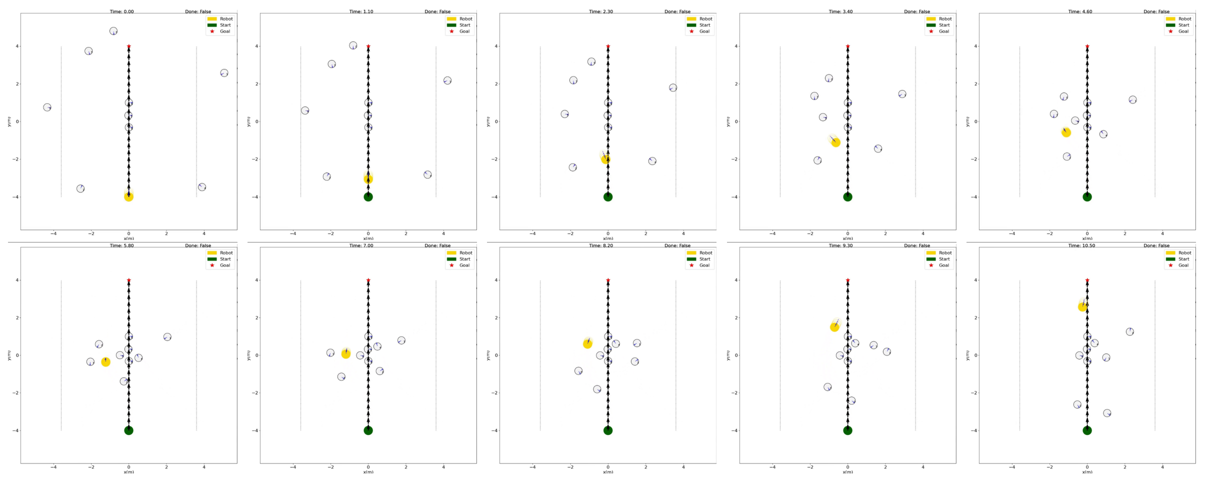

Figure 5.5: Real-to-sim Visual Episode. The order of frames goes from left to right and top to down. We can see the agent slowing down and turning to avoid collisions.

assumed—despite our empirical observation that the latent space tends to be dominated by path-tracking information when visualizing states by their distance in the latent space.

The qualitative behavior is also highly proficient. An example episode involving nine humans is shown in Figure 5.5. The robot's next three actions are visualized, illustrating its ability to slow down in cluttered areas. The robot also appropriately turns to avoid collisions while continuing to make progress toward the goal.

5.5 Simulation Experiments

To justify the results in the hardware experiments, we aim to replicate the results in simulation: train an expert in simulation using DRL and then use this expert to collect data for IL. In this case, we explicitly know the reward function the expert is trying to maximize against. As a result, we can evaluate our IL agents against the same reward function rather than using success rate as an all-encompassing performance proxy. It is important to note that a high success rate is a necessary but not sufficient condition for achieving a high average reward. Other navigation factors—such as path deviation and disturbance to humans—contribute to the reward but are not captured by the success rate alone.

If we can show that the IL agent with the full design achieves the highest reward, then we can likely say that the IL agent trained with real-world data is maximizing the reward function of the expert demonstrator which is complex and considers semantics of human movement to influence the expert's driving.

The training and testing scenarios are identical to the simulation testing in the hardware experiments. Because the training and testing scenarios are the same, we reduce the amount of training data. We evaluate both with 100k and 200k environment steps.

From Table 5.2, we observe that the expert achieves the highest average reward per episode, as expected. Comparing the 'Combined' model to the 'Base' model, we find that it performs better across both the 100k and 200k training settings. In the low-data regime of 100k demonstrations, the 'Combined' model achieves a 6x increase in reward relative to the 'Base' model. Even in the higher-data 200k setting, the 'Combined' model still maintains a roughly 2x improvement. As anticipated, models trained with more data outperform their counterparts trained with less, thus necessitating the importance of larger demonstration datasets.

Table 5.2: Sim-to-sim IL Results. Comparison of average reward and success rate across the DRL expert and IL variants.

Method	**Average Reward (↑)**	**Success Rate (↑)**
DRL	34.5	93.5
Base 100k	1.4	55.6
Combined 100k	8.8	61.1
Base 200k	7.6	71.3
Combined 200k	13.2	74.1

5.6 Summary

The main contribution of this chapter is the integration of several key design choices from the Machine Learning (ML) and IL literature, tailored to the specific problem of IL in social robot navigation. Through systematic ablations, we quantified the effect of each component, demonstrating that the full combination yields a substantial three-fold improvement in success rate when trained on real-world data. In the next section, we outline the major directions for future work necessary to advance this chapter toward publication.

5.7 Future Work

The first major direction for future work is to extend the simulation experiments to include all the ablations that were performed in the hardware experiments. Doing so would enable a more controlled and systematic analysis of each design choice in isolation on the reward, complementing the real-world results.

In this chapter, we did not assume a fixed corridor like in DR-MPC. This was intentional, as the environments encountered during data collection varied significantly—not only between different buildings but even within the same building, due to changes in hallway widths and static obstacles. As a result, it is not feasible to define a single corridor representation across all scenarios. However, in order to evaluate our model in more constrained real-world settings, such a component becomes necessary. Without it, we are limited to testing in open environments where the notion of staying within a corridor is either trivial or undefined.

We outline four potential workarounds for incorporating corridor constraints within our IL framework. The first is to post-process all collected rosbags and extract corridor boundaries using ray tracing. This would involve reconstructing the static environment and computing the closest obstacles to the left and right of the path. The second approach is to feed a 2D LiDAR scan of the environment directly into the model and have it learn to infer the corridor boundaries, following a strategy similar to SCAND [41]. However, neither of these two methods guarantees that the learned policy will avoid violating corridor constraints. A third option is to filter the model's output action paths by removing any that would result in a collision with the corridor. Finally, a fourth option is to switch to a simple heuristic controller when the robot nears a corridor boundary—for example, turning in place to face the reference path before returning control to the IL policy. Of these, the third and fourth methods are the most straightforward to implement and validate, and are thus the immediate next steps.

Another avenue for future work is the real-world validation of the 'Combined' model. We also plan to benchmark our framework against Nav2's MPPI controller [54]. We have already integrated T&R into the Nav2 stack, so it is ready for benchmarking. While Nav2's MPPI controller is inherently reactive, it is based on black-box optimization and can, in principle, incorporate motion forecasting to generate spatio-temporal costmaps to make it a more comprehensive baseline.

Lastly, we aim to benchmark against SCAND [41], a recent end-to-end BC approach that maps BEV point clouds directly to control actions. The data collection and processing pipelines for this comparison are already in place; once our BC framework is capable of handling static obstacles, we plan to train and evaluate the SCAND model within the same test environments.

Chapter 6

Conclusion and Future Directions

6.1 Conclusion

The goal of this thesis is to advance the field of social robot navigation through DRL and IL. The value of these learning-based approaches lies in their ability to learn directly from interaction—either with the world or from demonstration—thus avoiding the need to explicitly model human behavior, which remains a complex and unsolved problem. To this end, we contribute: (1) a simple decision process formulation that handles both static and dynamic obstacles, (2) a systems architecture for comprehensive, real-time state generation, (3) a sample-efficient DRL algorithm suitable for direct real-world training, and (4) an in-depth exploration of the design decisions critical to BC for social navigation.

First, we introduce a decision process formulation that unifies human avoidance and path tracking within a corridor-based abstraction. This representation accommodates both short and long-horizon navigation and generalizes to arbitrary environments by assuming the corridor is free of static obstacles. It avoids the jerky transitions often caused by waypoint transitions, while still allowing global planners to define the reference path. Learning is then focused purely on the dual objectives of path tracking and local human avoidance.

Next, we develop a robust, real-time system from perception to control using a single LiDAR sensor. The system detects and tracks humans using the LiDAR's reflectivity image and relies on the point cloud as input to T&R for long-term navigation. We design the architecture for efficiency, leveraging parallel processing and model optimization to maintain fast inference and system responsiveness in dynamic environments.

Then, we propose DR-MPC, a sample-efficient DRL method that can be trained directly in the real world. By isolating and placing an inductive prior on path tracking without sacrificing the performance of human avoidance, we significantly reduce the learning burden. Unlike classical DRL, which starts with random actions, or Residual DRL, which follows MPC in expectation, DR-MPC begins with MPC path tracking and gradually learns to balance between path following and human avoidance. We show that this strategy leads to more stable and efficient learning in both simulation and real-world deployments.

Lastly, we show that naïve applications of BC perform poorly in this domain. Through extensive exploration, we identify key architectural and training decisions that substantially improve performance. Our final approach achieves over a 3x improvement in success rate. Ongoing work includes validating these findings in simulation and deploying the improved model in the real world once corridor handling is in place.

Despite these contributions, several open challenges remain. Reducing the data requirements of DRL and IL is critical; semantically rich but compact representations could address this bottleneck. Most current approaches either use raw sensor inputs or coarse encodings of humans as circles.

6.2 Future Directions

A promising direction for future research in DRL and IL-based social robot navigation is the development of richer human representations—going beyond simple geometric abstractions like circles, but stopping short of raw sensory inputs, which often suffer from data inefficiency. One such representation is the human skeleton. In Figure 6.1, we show the output of RTM-Pose [39] applied to a reflectivity image, which extracts 17 keypoints per person. Compared to circle-based models, this skeleton representation offers substantially more semantic information. For instance, it enables the detection of individuals who are limping, distracted by their phones, or engaged in conversation.

These insights can directly inform policy behavior. A robot could, for example, give a wider berth to someone who is limping or inattentive, or route around a pair of people standing

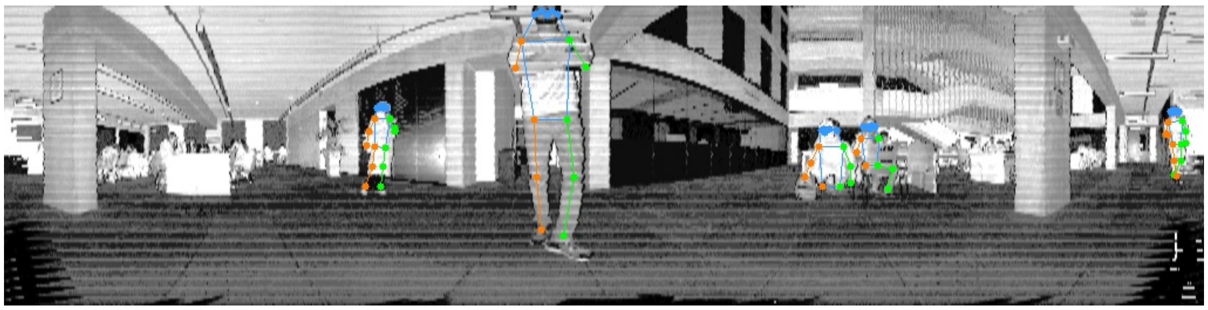

Figure 6.1: Skeleton Detection on Reflectivity Image. RTMPose being run on the reflectivity image. This particular model detects 17 keypoints as defined by the COCO dataset.

face-to-face. Conversely, if two individuals are standing with their backs to each other, the robot might safely pass between them.

While image-based representations can also capture these cues, they often include irrelevant details—such as texture, color, or background artifacts like glass walls or wood barriers—that may hinder generalization. Some works address this issue on the data side through domain randomization [81], but we propose a more principled approach that leverages domain knowledge to extract mid-level structured representations, like skeletons, for a more sample-efficient and interpretable alternative.

To validate this idea, we suggest a set of follow-up experiments and the pros and cons of each option.

Direct Reinforcement Learning. We considered training two agents with identical reward functions: one conditioned on a circle-based state representation and the other on a skeleton-based one. The hypothesis is that the richer semantic information provided by the skeletal representation will reduce state aliasing—ambiguities that arise when different scenarios look similar in the state space—leading to improved policy performance.

However, realistic human pose simulators remain immature. Synthetic human limbs often fail to capture the subtle cues—such as posture and gait—that are central to our hypothesis [42]. Consequently, the added noise from poorly rendered poses could degrade performance rather than enhance it.

An alternative is to reconstruct scenes from trajectory forecasting datasets and simulate a invisible robot navigation task where the robot avoids humans who do not react to its presence. Yet, few datasets simultaneously offer accurate skeleton annotations and reliable static obstacle maps [86]. Given these limitations—and our long-term goal of deploying on real robots interacting with dynamic human crowds—we believe the following offline IL approach is more viable.

Imitation Learning. Using a shared dataset, we can extract both circle and skeleton representations from identical scenes, train two separate policies, and compare their validation losses. A lower loss for the skeleton-conditioned policy would provide empirical support for our hypothesis. Qualitative trials on the real robot could then assess whether improvements in representation quality translate into tangible gains in safety and navigation efficiency.

We hope this thesis serves as a stepping stone toward learning-based social navigation systems that can operate fluently in homes, offices, and public spaces, behaving as naturally and socially compliantly as humans do.

Bibliography

[1] Robot Navigation in Crowds by Graph Convolutional Networks With Attention Learned From Human Gaze | IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/8990034.

[2] Joaquín Alori. Norfair: An open-source library for real-time multi-object tracking, 2020. Accessed: 2024-08-12.

[3] Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data, 2023.

[4] Changan Chen, Sha Hu, Payam Nikdel, Greg Mori, and Manolis Savva. Relational Graph Learning for Crowd Navigation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10007–10013, October 2020.

[5] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6015–6022, May 2019.

[6] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6015–6022, 2019.

[7] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu,

Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[8] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P. How. Socially aware motion planning with deep reinforcement learning. *CoRR*, abs/1703.08862, 2017.

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

[10] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P. How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 285–292, 2017.

[11] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024.

[12] Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

[13] Catie Cuan, Edward Lee, Emre Fisher, Anthony Francis, Leila Takayama, Tingnan Zhang, Alexander Toshev, and Sören Pirk. Gesture2path: Imitation learning for gesture-aware navigation, 2022.

[14] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning, 2019.

[15] Taylor Denouden, Rick Salay, Krzysztof Czarnecki, Vahdat Abdelzad, Buu Phan, and Sachin Vernekar. Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance. *arXiv preprint arXiv:1812.02765*, 2018.

[16] Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, August 2024.

[17] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

[18] Gabriel Dulac-Arnold, Natasha Levine, Daniel J. Mankowitz, Todd Hester, Timothy Lillicrap, and Ethan Dyer. Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*, 110:2419–2468, September 2021.

[19] Michael Everett, Yu Fan Chen, and Jonathan P. How. Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3052–3059, October 2018.

[20] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning, 2021.

[21] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, March 1997.

[22] Anthony Francis, Claudia Pérez-D'Arpino, Chengshu Li, Fei Xia, Alexandre Alahi, Rachid Alami, Aniket Bera, Abhijat Biswas, Joydeep Biswas, Rohan Chandra, Hao-Tien Lewis Chiang, Michael Everett, Sehoon Ha, Justin Hart, Jonathan P. How, Haresh Karnan, Tsang-Wei Edward Lee, Luis J. Manso, Reuth Mirsky, Sören Pirk, Phani Teja Singamaneni, Peter Stone, Ada V. Taylor, Peter Trautman, Nathan Tsoi, Marynel Vázquez, Xuesu Xiao, Peng Xu, Naoki Yokoyama, Alexander Toshev, and Roberto Martín-Martín. Principles and guidelines for evaluating social robot navigation algorithms. *J. Hum.-Robot Interact.*, 14(2), February 2025.

[23] Scott Fujimoto, Herke Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1587–1596. PMLR, July 2018.

[24] Paul Furgale and Timothy D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 2010.

[25] Paul Furgale and Timothy D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 2010.

[26] Alessandro Giusti, Jérôme Guzzi, Dan C. Cireşan, Fang-Lin He, Juan P. Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, Davide Scaramuzza, and Luca M. Gambardella. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.

[27] Christian Gloor. Pedsim: Pedestrian crowd simulation. *URL http://pedsim. silmaril. org*, 5(1), 2016.

[28] Silvia Guillén-Ruiz, Juan Pedro Bandera, Alejandro Hidalgo-Paniagua, and Antonio Bandera. Evolution of Socially-Aware Robot Navigation. *Electronics*, 12(7):1570, January 2023.

[29] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870. PMLR, July 2018.

[30] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[31] Mahmoud Hamandi, Mike D'Arcy, and Pooyan Fazli. Deepmotion: Learning to navigate like humans. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–7, 2019.

[32] James R. Han, Hugues Thomas, Jian Zhang, Nicholas Rhinehart, and Timothy D. Barfoot. Dr-mpc: Deep residual model predictive control for real-world social navigation. *IEEE Robotics and Automation Letters*, 10(4):4029–4036, 2025.

[33] Katherine Hermann, Hossein Mobahi, Thomas FEL, and Michael Curtis Mozer. On the foundations of shortcut learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[34] Noriaki Hirose, Catherine Glossop, Ajay Sridhar, Dhruv Shah, Oier Mees, and Sergey Levine. Lelan: Learning a language-conditioned navigation policy from in-the-wild videos, 2024.

[35] Noriaki Hirose, Dhruv Shah, Ajay Sridhar, and Sergey Levine. Sacson: Scalable autonomous control for social navigation. *IEEE Robotics and Automation Letters*, 9(1):49–56, 2024.

[36] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4572–4580, Red Hook, NY, USA, 2016. Curran Associates Inc.

[37] Kevin Huang, Rwik Rana, Alexander Spitzer, Guanya Shi, and Byron Boots. DATT: Deep Adaptive Trajectory Tracking for Quadrotor Control, October 2023.

[38] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025.

[39] Tao Jiang, Peng Lu, Li Zhang, Ningsheng Ma, Rui Han, Chengqi Lyu, Yining Li, and Kai Chen. Rtmpose: Real-time multi-person pose estimation based on mmpose, 2023.

[40] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[41] Haresh Karnan, Anirudh Nair, Xuesu Xiao, Garrett Warnell, Sören Pirk, Alexander Toshev, Justin Hart, Joydeep Biswas, and Peter Stone. Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation. *IEEE Robotics and Automation Letters*, 7(4):11807–11814, 2022.

[42] Linh Kästner, Volodymyir Shcherbyna, Huajian Zeng, Tuan Anh Le, Maximilian Ho-Kyoung Schreff, Halid Osmaev, Nam Truong Tran, Diego Diaz, Jan Golebiowski, Harold Soh, et al. Arena 3.0: Advancing social navigation in collaborative and highly dynamic environments. *arXiv preprint arXiv:2406.00837*, 2024.

[43] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[44] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025.

[45] Polina Kirichenko, Pavel Izmailov, and Andrew G Wilson. Why normalizing flows fail to detect out-of-distribution data. *Advances in neural information processing systems*, 33:20578–20589, 2020.

[46] Marina Kollmitz, Torsten Koller, Joschka Boedecker, and Wolfram Burgard. Learning human-aware robot navigation from physical interaction via inverse reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11025–11031, 2020.

[47] Linh KU+000E4stner, Johannes Cox, Teham Buiyan, and Jens Lambrecht. All-in-One: A DRL-based Control Switch Combining State-of-the-art Navigation Planners. In *2022*

International Conference on Robotics and Automation (ICRA), pages 2861–2867, May 2022.

[48] Bo Ling, Yan Lyu, Dongxiao Li, Guanyu Gao, Yi Shi, Xueyong Xu, and Weiwei Wu. Socialgail: Faithful crowd simulation for social robot navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16873–16880, 2024.

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

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

[51] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. *CoRR*, abs/1709.10082, 2017.

[52] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[53] Brandon Luders, Georges Aoude, Joshua Joseph, Nicholas Roy, and Jonathan How. Probabilistically safe avoidance of dynamic obstacles with uncertain motion patterns. 07 2011.

[54] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[55] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[56] Sango Matsuzaki and Yuji Hasegawa. Learning crowd-aware robot navigation from challenging environments via distributed deep reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4730–4736, 2022.

[57] Christoforos I. Mavrogiannis, Francesca Baldini, Allan Wang, Dapeng Zhao, Pete Trautman, Aaron Steinfeld, and Jean Oh. Core challenges of social robot navigation: A survey. *CoRR*, abs/2103.05668, 2021.

[58] Siddarth Narasimhan, Aaron Hao Tan, Daniel Choi, and Goldie Nejat. Olivia-nav: An online lifelong vision language approach for mobile robot social navigation, 2025.

[59] Duc M. Nguyen, Mohammad Nazeri, Amirreza Payandeh, Aniket Datar, and Xuesu Xiao. Toward human-like social robot navigation: A large-scale, multi-modal, social human navigation dataset. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7442–7447, 2023.

[60] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

[61] Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4045–4054. PMLR, 10–15 Jul 2018.

[62] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[63] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268, 2009.

[64] Ashwini Pokle, Roberto Martín-Martín, Patrick Goebel, Vincent Chow, Hans M. Ewald, Junwei Yang, Zhenkai Wang, Amir Sadeghian, Dorsa Sadigh, Silvio Savarese, and Marynel Vázquez. Deep local trajectory replanning and control for robot navigation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5815–5822, 2019.

[65] Dean A. Pomerleau. Alvinn: an autonomous land vehicle in a neural network. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, NIPS'88, page 305–313, Cambridge, MA, USA, 1988. MIT Press.

[66] Noé Pérez-Higueras, Roberto Otero, Fernando Caballero, and Luis Merino. Hunavsim: A ros 2 human navigation simulator for benchmarking human-aware robot navigation. *IEEE Robotics and Automation Letters*, 8(11):7130–7137, 2023.

[67] Lei Qin, Zefan Huang, Chen Zhang, Hongliang Guo, Marcelo Ang, and Daniela Rus. Deep imitation learning for autonomous navigation in dynamic pedestrian environments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4108–4115, 2021.

[68] Sepehr Samavi, James R. Han, Florian Shkurti, and Angela P. Schoellig. Sicnav: Safe and interactive crowd navigation using model predictive control and bilevel optimization, 2024.

[69] Sunil Srivatsav Samsani and Mannan Saeed Muhammad. Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):5223–5230, 2021.

[70] Max Schwarzer, Johan Obando-Ceron, Aaron Courville, Marc Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency, 2023.

[71] Jordy Sehn, Timothy D. Barfoot, and Jack Collier. Off the beaten track: Laterally weighted motion planning for local obstacle avoidance. *IEEE Transactions on Field Robotics*, 1:249–275, 2024.

[72] Samaneh Hosseini Semnani, Hugh Liu, Michael Everett, Anton De Ruiter, and Jonathan P How. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3221–3226, 2020.

[73] Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. The pitfalls of imitation learning when actions are continuous, 2025.

[74] Mario Srouji, Hugues Thomas, Yao-Hung Hubert Tsai, Ali Farhadi, and Jian Zhang. Safer: Safe collision avoidance using focused and efficient trajectory search with reinforcement learning. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–8. IEEE, 2023.

[75] Chao Sun, Xing Wu, Yanxu Su, Xiasheng Shi, and Changyin Sun. Multithreaded asynchronous deep reinforcement learning with multisensor fusion for robot collision avoidance. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2025.

[76] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *International Conference on Machine Learning*, pages 20827–20840. PMLR, 2022.

[77] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*.

[78] Lei Tai, Jingwei Zhang, Ming Liu, and Wolfram Burgard. Socially compliant navigation through raw depth inputs with generative adversarial imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1111–1117, 2018.

[79] Lei Tai, Jingwei Zhang, Ming Liu, and Wolfram Burgard. Socially compliant navigation through raw depth inputs with generative adversarial imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1111–1117, 2018.

[80] Hugues Thomas, Jian Zhang, and Timothy D. Barfoot. The Foreseeable Future: Self-Supervised Learning to Predict Dynamic Scenes for Indoor Navigation. *IEEE Transactions on Robotics*, pages 1–19, 2023.

[81] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world, 2017.

[82] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 797–803, 2010.

[83] Nathan Tsoi, Alec Xiang, Peter Yu, Samuel S. Sohn, Greg Schwartz, Subashri Ramesh, Mohamed Hussein, Anjali W. Gupta, Mubbasir Kapadia, and Marynel Vázquez. Sean 2.0: Formalizing and generating social situations for robot navigation. *IEEE Robotics and Automation Letters*, pages 1–8, 2022.

[84] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-Body Collision Avoidance. In Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger, editors, *Robotics Research*, Springer Tracts in Advanced Robotics, pages 3–19, Berlin, Heidelberg, 2011. Springer.

[85] Ashish Vaswani. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[86] Edward Vendrow, Duy Tho Le, Jianfei Cai, and Hamid Rezatofighi. Jrdb-pose: A large-scale dataset for multi-person pose estimation and tracking, 2023.

[87] Weizheng Wang, Ike Obi, Aniket Bera, and Byung-Cheol Min. Unifying large language model and deep reinforcement learning for human-in-loop interactive socially-aware navigation, 2025.

[88] Weizheng Wang, Ike Obi, and Byung-Cheol Min. Multi-agent llm actor-critic framework for social robot navigation, 2025.

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

[90] Zaitian Wang, Pengfei Wang, Kunpeng Liu, Pengyang Wang, Yanjie Fu, Chang-Tien Lu, Charu C. Aggarwal, Jian Pei, and Yuanchun Zhou. A comprehensive survey on data augmentation, 2025.

[91] Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories, 2020.

[92] Fei Xia, William B. Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020.

[93] Jing Xie, Yongjun Zhang, Huanhuan Yang, Qianying Ouyang, Fang Dong, Xinyu Guo, Songchang Jin, and Dianxi Shi. Crowd perception communication-based multi- agent path finding with imitation learning. *IEEE Robotics and Automation Letters*, 9(10):8929–8936, 2024.

[94] Zhanteng Xie, Pujie Xin, and Philip Dames. Towards safe navigation through crowded dynamic environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4934–4940, 2021.

[95] Yigit Yildirim and Emre Ugur. Learning social navigation from demonstrations with conditional neural processes. *Interaction Studies*, 23(3):427–468, December 2022.

[96] Hui Zeng, Rong Hu, Xiaohui Huang, and Zhiying Peng. Robot navigation in crowd based on dual social attention deep reinforcement learning. *Mathematical Problems in Engineering*, 2021(1):7114981, 2021.

[97] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020.

[98] Haibin Zhou, Zichuan Lin, Junyou Li, Qiang Fu, Wei Yang, and Deheng Ye. Revisiting discrete soft actor-critic. *arXiv preprint arXiv:2209.10081*, 2022.

[99] Yanying Zhou and Jochen Garcke. Learning crowd behaviors in navigation with attention-based spatial-temporal graphs. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5485–5491, 2024.