

Learning Human-Aware Strategies for Legible and Predictable Robot Navigation

by

Anna Moskalenko

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2025

© Anna Moskalenko 2025

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

A challenge in human-robot interaction, particularly in dynamic and crowded environments, is to design navigation that is both legible and predictable. For people to feel comfortable in the same environment as robots, their movements must be legible - ensuring quick understanding of the robot’s intentions - and predictable, meaning they align with human expectations.

In this thesis, we introduce a learning-based navigation system that leverages a vector reward function to capture the dual objectives of legibility and predictability. Rather than relying on manually designed transitioning rules or fixed weighting parameters such as α and β , the reward function is learned from expert demonstrations generated by a planner (LPSNav) that blends between these objectives using a continuous parameter. Our approach does not attempt to replicate the planner’s handcrafted logic, but instead generalizes the emergent patterns in its trajectories through a supervised learning framework inspired by the structure of maximum entropy IRL.

The proposed architecture includes:

- an LSTM-based Robot Motion History Encoder,
- a CNN-based Environment Encoder,
- an LSTM-based Human History Encoder,
- and a two-channel Reward Analysis Engine.

The system was evaluated both in simulation using the nuScenes dataset and in real-world trials using the Clearpath Jackal robot. In the user study, participants interacted with the robot in predefined navigation scenarios, and their feedback was used to assess the perceived clarity and predictability of the robot’s actions.

Results show that our method generates more human-like trajectories and outperforms baseline models in scenarios requiring socially acceptable motion, such as intersections, sidestepping, and close-proximity passing. We provide both quantitative metrics (e.g., ADE and FDE) and qualitative visualizations demonstrating smooth and socially-aware navigation behavior.

This work represents a step toward safer and more intuitive human-robot coexistence, offering a practical solution for real-world robotic deployment.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Stephen L. Smith, for introducing me to this fascinating topic. His guidance, patience, and insightful feedback have been invaluable throughout every stage of this research. Working under his supervision has been both a challenging and deeply rewarding academic journey.

I am also sincerely thankful to all members of the Smith Autonomy Lab for their constructive comments, engaging discussions, and encouragement during the development of this work.

To my friends - thank you for your unwavering support, understanding, and belief in me, especially during the most stressful moments. Your encouragement helped me stay focused and motivated when it mattered most.

I extend my heartfelt thanks to my family. Your constant love, strength, and support have been the foundation of everything I do. This achievement would not have been possible without you.

Finally, I would also like to acknowledge the use of OpenAI's ChatGPT language model, which was employed for grammar correction, stylistic refinement, and academic rephrasing of selected passages in this thesis. All ideas, analyses, and conclusions presented are entirely my own.

Dedication

This thesis is dedicated to my parents and all other family members.

Table of Contents

Author's Declaration	ii
Abstract	iii
Acknowledgements	iv
Dedication	v
List of Figures	x
List of Tables	xii
1 Introduction	1
1.1 Thesis Contributions	3
1.2 Thesis Outline	5
2 Literature Review	7
2.1 Foundations of Socially-Aware Robot Navigation	7
2.2 Rule-Based Social Navigation	9
2.3 Learning-Based Approaches to Social Navigation	11
2.4 Smooth Maximum Entropy Deep Inverse Reinforcement Learning (S-MEDIRL)	12
2.5 Identified Research Gap	13

3	Preliminaries	15
3.1	Problem Formulation	15
3.2	Markov Decision Process Formulation	16
3.3	Trajectory and Interaction Modeling	17
3.4	Legibility and Predictability Definitions	18
3.5	Assumptions and Environment Setup	19
4	Proposed Method	20
4.1	System Overview	20
4.2	Motion History Encoder (MHE) with LSTM Model	22
4.2.1	Data Preparation and Preprocessing	22
4.2.2	LSTM Model Architecture	23
4.2.3	Model Training	26
4.2.4	Evaluation and Results	27
4.2.5	Discussion of MHE Contribution	30
4.3	Environment Encoder (CNN)	30
4.3.1	Purpose and Context	31
4.3.2	Input Data: Occupancy Grid from LiDAR	31
4.3.3	CNN Architecture and Image Processing Pipeline	32
4.3.4	Training Methodology	34
4.3.5	Qualitative Results and Output Interpretation	35
4.3.6	Contribution to Navigation	35
4.3.7	Discussion of Environment Encoder Contribution	36
4.4	Human Trajectory Forecasting Using LSTM and CVAE	37
4.4.1	Input Modalities and Preprocessing	38
4.4.2	LSTM-Based Human Encoder	40
4.4.3	Modeling Inter-Agent Interaction	41
4.4.4	Multimodal Future Prediction via CVAE	43

4.4.5	Training Setup	45
4.4.6	Results and Analysis	46
4.4.7	Discussion of Human Trajectory Encoder Contribution	48
4.5	Reward Function and Learning Framework	49
4.5.1	Reward Function Definition	49
4.5.2	Network Architecture	50
4.5.3	Training with Pseudo-Supervision	52
4.5.4	Incorporating S-MEDIRL-Inspired Regularization	53
4.5.5	Context-Aware Reward Aggregation	54
4.5.6	Evaluation and Interpretation	56
4.5.7	Discussion	59
4.6	Trajectory Selection and Planning	59
4.6.1	Trajectory Generation via LPSNav	59
4.6.2	Reward Evaluation	60
4.6.3	Integration with Navigation System	60
5	Experimental Evaluation	62
5.1	Experimental Setup	62
5.2	Evaluation Methodology	63
5.3	Results	64
5.3.1	Task Load	64
5.3.2	Trust, Comfort, and Legibility	65
5.3.3	Summary	67
5.4	Comparison with Baseline	67
5.4.1	Qualitative Comparison with CADRL	68
5.4.2	Quantitative Comparison Based on Trajectory Logs	70
5.4.3	Comparison with LPSNav (Expert Planner)	72

6 Conclusion	74
6.1 Limitations	75
6.2 Future Work	75
References	77

List of Figures

1.1	Robot behaviors: standard vs socially-aware	2
1.2	System architecture overview	5
2.1	Legible vs predictable trajectories	8
4.1	Proposed system architecture	21
4.2	2D training trajectories (nuScenes)	23
4.3	Motion History Encoder (MHE)	25
4.4	LSTM training loss	27
4.5	Arc-shaped trajectory	29
4.6	Local turns & variations	29
4.7	Sine-like trajectory	29
4.8	Linear diagonal trajectory	29
4.9	Predicted vs actual trajectories	29
4.10	LiDAR to occupancy grid	32
4.11	Environment Encoder (CNN)	33
4.12	Real vs predicted occupancy grid	36
4.13	Human Encoder architecture	37
4.14	Camera-based pedestrian perception	38
4.15	RGB–LiDAR fusion for localization	39
4.16	Inter-agent interaction graph	43

4.17 Pedestrian trajectories (training set)	47
4.18 Multimodal trajectory predictions	48
4.19 Reward network architecture	51
4.20 Context-aware trajectory	55
4.21 Scalarization weights over time	56
4.22 Predicted reward components	58
5.1 Human walking head-on	63
5.2 Human walking diagonally	63
5.3 Human walking in parallel	63
5.4 Interaction scenarios overview	63
5.5 Task Load Ratings for LPSNav+Reward.	65
5.6 Task Load Ratings for CADRL	65
5.7 Trust and Predictability Ratings – LPSNav+Reward	66
5.8 Trust and Predictability Ratings – CADRL	66
5.9 Comfort and Safety Ratings – LPSNav+Reward	66
5.10 Comfort and Safety Ratings – CADRL	66
5.11 Legibility and Performance Ratings – LPSNav+Reward	67
5.12 Legibility and Performance Ratings – CADRL	67
5.13 CADRL vs LPSNav behaviors	69
5.14 Avoidance behavior: CADRL vs LPSNav	71

List of Tables

2.1	Comparison of Rule-Based and Learning-Based Social Navigation Systems	10
4.1	Sample reward values for selected trajectories.	57
5.1	Qualitative comparison between CADRL and LPSNav.	68
5.2	Quantitative comparison of CADRL and LPSNav+Reward based on real-world trajectory data.	70
5.3	Quantitative comparison of LPSNav (expert planner) and LPSNav + Re- ward (ours) based on trajectory data.	72

Chapter 1

Introduction

As mobile robots become more common in public and private spaces, their ability to navigate in a way that is not only safe and efficient, but also socially understandable, is becoming increasingly important. While classical planners focus on optimizing geometric properties such as path length or obstacle avoidance, these approaches often fall short in human-rich environments where expectations, intentions, and social cues play a central role [10].

In human-robot interaction (HRI), two critical properties have emerged for socially-aware navigation: *legibility* and *predictability*. Legibility refers to how clearly a robot's motion communicates its immediate goal or intention, allowing an observer to quickly infer its purpose. In contrast, predictability concerns the alignment of the robot's trajectory with the human's expectations based on context [2].

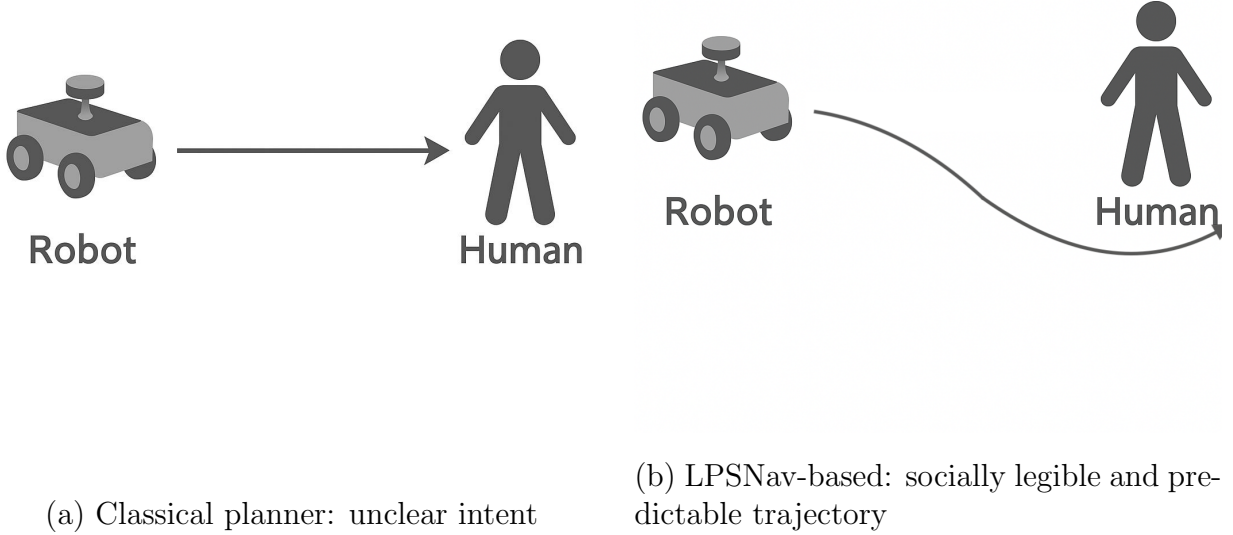


Figure 1.1: Comparison of robot behaviors in human environments. Left: standard navigation induces uncertainty. Right: socially-aware motion is clearer and easier to interpret.

Algorithms such as LPSNav (Legibility-Predictability-Social Navigation) [2] attempt to incorporate these qualities through a continuous transition between objectives: early portions of a path are generated to be legible using predefined motion templates, while later segments gradually prioritize predictability. Although this hybrid strategy shows promise, it relies on manually designed heuristics and hand-tuned switching logic, which limits adaptability in environments with diverse or dynamic agent behavior. In highly dynamic settings - such as intersections, dense traffic scenes, or spaces with complex interactions - these heuristics may break down or fail to generalize.

This thesis proposes an alternative to rule-based strategies by introducing a learning-based framework that integrates Smooth Maximum Entropy Deep Inverse Reinforcement Learning (S-MEDIRL) [29] into the LPSNav planning pipeline. Rather than relying on predefined switching heuristics or handcrafted cost functions, S-MEDIRL [29, 64] learns a vector-valued reward function from trajectory-level expert demonstrations.

We acknowledge that these expert demonstrations originate from LPSNav, but our goal is not to imitate its internal logic - rather, to learn generalizable behavior that captures the

emergent effects of LPSNav trajectories while removing dependence on manually specified transitions between legibility and predictability.

By training this reward function on expert trajectories derived from realistic urban scenes such as nuScenes [3], the robot can dynamically adapt its navigation strategy to context, selecting trajectories that best balance human interpretability and goal alignment - without the need for manual tuning of switching parameters.

To validate the proposed approach, we conduct a two-part evaluation: (1) simulation-based experiments using the **nuScenes** dataset, which provides dense, multi-agent urban interaction data; and (2) a user study involving real human participants observing the robot in various planning scenarios. In both cases, we assess performance across quantitative metrics (such as displacement error) and subjective ratings of perceived legibility, predictability, and comfort [9].

Ultimately, this work contributes to bridging the gap between algorithmic navigation and human-centered design by enabling robots to generate trajectories that dynamically align with human expectations, without relying on handcrafted switching rules.

1.1 Thesis Contributions

The following are the key contributions of this thesis:

1. **Synergistic encoder architecture for decomposable social reward learning.** Rather than focusing on individual encoder design, our contribution lies in the synergistic integration of three specialized modules that collectively enable learning a decomposable representation of social reward signals. This modular design supports interpretability, reusability, and disentangled reasoning over motion, environment, and social expectations:
 - 1.1 A Motion History Encoder (LSTM-based) that captures temporal dynamics of the robot’s past trajectory and predicts its short-term future motion [37].
 - 1.2 An Environment Encoder (CNN-based) that extracts spatial context from occupancy grids for obstacle and free-space awareness [44].
 - 1.3 A Human Expectation Encoder (LSTM-based, optionally GNN-augmented) that models human trajectory history and predicts expectations about future robot behavior [58].

2. **Vector-valued reward function capturing both legibility and predictability.**
 Unlike traditional IRL methods that produce a scalar reward [64], we formulate a multi-objective reward function with explicit components:

$$\phi_t = [h_t \parallel e_t \parallel \hat{y}_t \parallel v_t^{\text{norm}}]$$

where each term represents a key aspect of motion clarity and contextual compliance. This design enables the S-MEDIRL framework [29] to infer reward weights without hand-tuned α/β trade-offs.

3. **Integration with LPSNav-based expert demonstrations.**

To train the reward function, we generated a dataset of expert trajectories using the rule-based LPSNav planner [2]. We use these trajectories not to replicate heuristic switching behavior, but to learn an interpretable and generalizable reward signal.

4. **Improved human perception in real-world settings.**

User study results demonstrate that the proposed system is perceived as significantly more predictable, legible, and comfortable than baseline planners such as CADRL [13] and Social Force Models in real navigation scenarios [24].

5. **Superior performance in socially critical scenarios.**

Quantitative analysis reveals that the proposed system initiates avoidance maneuvers earlier and maintains a larger, more socially appropriate distance to nearby humans compared to CADRL [13] and other baselines, particularly in scenarios involving head-on, diagonal, and close-proximity interactions [13, 24].

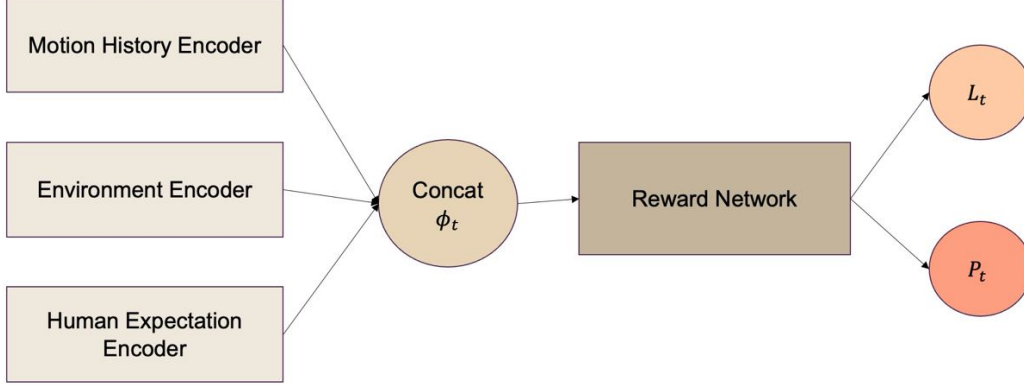


Figure 1.2: Overall system architecture showing the flow from modular encoders to the reward function and final output policy. The reward function is trained in a supervised manner using pseudo-ground-truth legibility and predictability signals derived from LP-SNav, with smoothness regularization and dynamic weighting inspired by the S-MEDIRL framework.

Together, these contributions advance the field of human-aware robot navigation by enabling legible and predictable motion without relying on hand-crafted behavioral rules.

1.2 Thesis Outline

Chapter 2 reviews prior work on human-aware navigation, including both rule-based systems and learning-based approaches. It discusses the role of legibility and predictability in social motion planning and highlights the gap in existing methods that motivate this thesis.

Chapter 3 formalizes the problem setting, introduces the Markov Decision Process (MDP) formulation, and defines legibility and predictability using probabilistic inference. It also outlines the input modalities available to the robot and the structure of the feature representations used in learning.

Chapter 4 presents the full proposed system. This includes all models used in the navigation pipeline:

- the **Motion History Encoder (MHE)** based on LSTM, which models the robot’s

recent trajectory;

- the **Environment Encoder**, a CNN that processes LiDAR-based occupancy grids;
- the **Human Expectation Encoder**, which predicts future human motion using CVAE;
- the **Reward Function**, a vector-valued formulation capturing legibility and predictability;
- and the **Trajectory Selection and Planning** logic that integrates reward-based evaluation.

We explain the training methodology for each component and describe how the S-MEDIRL framework is used to optimize the reward function using expert demonstrations from LPSNav.

Chapter 5 presents the experimental evaluation. We report simulation results on the nuScenes dataset using metrics such as ADE and FDE, and analyze results from a real-world user study conducted with the Clearpath Jackal robot. The study focuses on human perception of clarity, predictability, and safety in social navigation scenarios.

Finally, Chapter 6 summarizes the findings, discusses current limitations - such as reliance on offline learning and dataset-specific priors - and outlines future directions, including real-time adaptation, online reward refinement, and deployment in more complex multi-agent environments.

Chapter 2

Literature Review

We begin with an overview of the foundational work in socially-aware robot navigation and the emergence of legibility and predictability as key concepts in Section 2.1. Section 2.2 focuses on social navigation frameworks, including rule-based systems like LPSNav, and highlights their reliance on hand-crafted heuristics. Section 2.3, we summarize recent learning-based approaches to trajectory prediction and social compliance, including inverse reinforcement learning (IRL) and deep generative models. Section 2.4 reviews the Smooth Maximum Entropy Deep Inverse Reinforcement Learning (S-MEDIRL) framework, which introduces spatially smoothed reward learning for social navigation. Finally, Section 2.5, we identify the gap in existing literature regarding the joint learning of legibility and predictability, which motivates the development of the method proposed in this thesis.

2.1 Foundations of Socially-Aware Robot Navigation

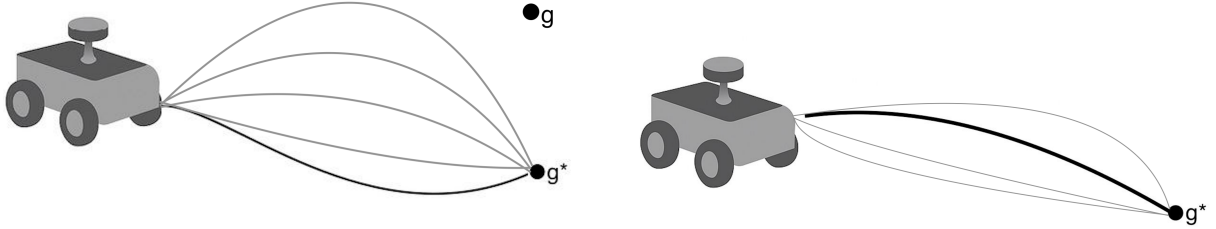
Socially-aware robot navigation focuses on enabling robots to move through environments populated by humans in ways that are not only safe and efficient, but also interpretable by human observers. Traditional motion planners often optimize geometric metrics such as path length or obstacle clearance, but fail to consider whether their motion aligns with human expectations or communicates their intent effectively. This disconnect may result in undesirable behaviors like sudden swerves, hesitation, or the so-called “freezing robot” problem [55]. This problem has led to increased interest in socially-aware navigation frameworks that account for human perception and expectations [50, 28].

Recent work introduces two key properties of motion that help address this challenge: *legibility* and *predictability* [10, 2]. Legible motion is defined as *intent-expressive*, meaning

that the robot’s trajectory allows observers to quickly and confidently infer its intended collision avoidance strategy. In contrast, predictable motion is defined as *expectation-consistent*, meaning that it aligns with what an observer would naturally anticipate based on context and social conventions. Recent survey efforts have further emphasized these properties as essential for natural and effective human-robot interaction [28].

In dynamic environments where the robot’s final goal is not known to surrounding humans, the notion of legibility is redefined not in terms of the global goal, but with respect to the robot’s intended interaction outcome - for example, which side it plans to pass on [2]. Thus, a trajectory is legible if it reduces ambiguity early in the interaction, while predictable trajectories reinforce expectations once that intent has been established.

These properties are formally modeled using probabilistic inference functions. Legibility is captured by an inference function $f_L(\xi)$ that estimates the robot’s interaction goal given a partial trajectory [10], whereas predictability is modeled by $f_P(g)$, which estimates the most likely trajectory given a known interaction goal. The legibility of a motion increases if f_L correctly infers the robot’s avoidance strategy early, while predictability increases if f_P returns a trajectory close to the robot’s actual motion. This distinction is illustrated Figure 2.1, which shows how legible and predictable trajectories differ in terms of early goal inference versus alignment with expected motion.



(a) Legible trajectory: early deviation to express intent.

(b) Predictable trajectory: minimal deviation, smooth path.

Figure 2.1: Comparison of legible and predictable robot trajectories toward the same goal g^* .

A socially-aware planner must therefore reason about the trade-off between legibility and predictability throughout the interaction. In ambiguous scenarios - such as head-on

encounters - legibility should be prioritized to clearly signal intent. Once the passing side is inferred by the human observer, the planner should shift its focus to predictable motion, thereby allowing smooth and confident cooperation.

This framework motivates the need for dynamic goal regions rather than fixed goal points, and it underpins the design of planners such as LPSNav, which explicitly model interaction goals like passing on the left or right as latent variables and adapt their motion accordingly [2].

2.2 Rule-Based Social Navigation

Early efforts to enable socially appropriate robot navigation have relied heavily on rule-based methods. These systems define a fixed set of heuristics or behaviors that the robot executes when interacting with humans, often encoded as explicit “if-then” rules or priority-based policies. Although limited in flexibility, rule-based systems offer transparency, low computational cost, and guaranteed behavioral bounds, which make them appealing for safety-critical environments [51, 38].

A representative example of this class is the Legibility-Predictability Social Navigation (LPSNav) planner, which models robot behavior as a combination of discrete interaction goals (e.g., “pass on the left”) and continuous trajectory optimization [2]. Instead of reasoning about goal positions alone, LPSNav introduces the notion of latent *interaction goals*, which refer to the robot’s intended outcome in terms of spatial relation to the human. These goals are predefined and used to generate distinct avoidance behaviors that are either legible or predictable depending on context.

For instance, when encountering a pedestrian head-on, LPSNav selects an avoidance goal such as “pass on the right” and optimizes the robot’s trajectory to align with this intent. If ambiguity is high - such as in frontal encounters with unclear lateral displacement - the planner prioritizes legibility by diverging early and explicitly in the chosen direction. Conversely, once the avoidance side is inferred, the planner prioritizes predictability by maintaining a smooth and consistent path.

Rule-based systems like LPSNav explicitly separate legibility and predictability and can transition between them depending on the scenario. However, these systems are limited by their reliance on handcrafted parameters, such as fixed lateral offset distances, thresholds for intent switching, and context-specific rule activation. They also require prior knowledge of interaction types and fail to generalize well in unfamiliar or unstructured environments [38].

Furthermore, rule-based planners often lack the ability to adapt online to nuanced human behaviors, such as hesitation, yielding, or last-moment trajectory changes. While they perform well in idealized or controlled settings, their performance deteriorates in highly dynamic multi-agent environments where social compliance must emerge from learned experience rather than static definitions.

These limitations, as highlighted in recent reviews [50], motivate the exploration of learning-based approaches that can adapt to human social cues in real time, generalize to novel situations, and model legibility and predictability as emergent properties of interaction.

Aspect	Rule-Based	Learning-Based
Behavior Definition	Explicitly defined via heuristics or control rules (e.g., “always pass on the right”)	Learned from expert trajectories using imitation learning or inverse reinforcement learning (IRL)
Adaptability	Limited to predefined conditions	High, can generalize to unseen scenarios if trained on diverse data
Context Sensitivity	Hard-coded responses to limited context (e.g., proximity)	Emerges from data; adapts based on context such as crowd density or human motion patterns
Legibility and Predictability	Can be explicitly encoded (e.g., through motion primitives or social rules)	Often implicit in reward; requires multi-objective formulation to decouple (e.g., vector-valued reward in S-MEDIRL)
Interpretability	High - easy to trace decisions back to rules	Moderate - decisions may be opaque, especially with deep neural networks
Need for Demonstrations	Not required	Required - expert trajectories are needed for training
Real-World Robustness	Fails under unexpected behavior or noisy environments	Dependent on generalization; robust if trained with diverse and representative data

Table 2.1: Comparison of Rule-Based and Learning-Based Social Navigation Systems

2.3 Learning-Based Approaches to Social Navigation

To overcome the limitations of rule-based systems, recent research in social robot navigation has increasingly adopted learning-based approaches. These methods leverage demonstration data to model human navigation patterns and derive behavior policies that generalize to dynamic, multi-agent environments [47]. Broadly, these approaches fall into two main categories: (i) trajectory forecasting models, which predict human motion to enable proactive planning [1, 58], and (ii) reinforcement learning (RL) and inverse reinforcement learning (IRL) methods, which directly learn policies or reward functions from expert demonstrations [64, 29]. Recent hybrid architectures, such as NaviSTAR [60], combine spatio-temporal graph attention with preference learning to model both crowd dynamics and individual behavior preferences, further improving social navigation in dense environments.

Trajectory Forecasting Models

Trajectory forecasting models focus on predicting the future motion of nearby humans, typically by conditioning on past trajectories and environmental context. Social LSTM [1] and its successors introduced the idea of modeling human-human interactions using recurrent neural networks with spatial pooling. Later works, such as Social GAN [23], and Trajectron++ [49], further improved multi-agent prediction by incorporating generative modeling, attention mechanisms, and semantic scene context.

These models often use conditional variational autoencoders (CVAEs) to represent the multimodal nature of human motion. Given the same history, multiple plausible futures may exist (e.g., pass on the left or the right), and CVAE-based models can sample from this distribution to provide diverse, socially plausible predictions. While these models are not navigation planners per se, they serve as critical components for socially aware motion planning by anticipating human behavior and avoiding future conflicts.

Learning Rewards through IRL

Another branch of work focuses on learning reward functions that capture human-like behavior through inverse reinforcement learning. IRL techniques aim to recover a reward function under which expert demonstrations are optimal. This allows robots to not just imitate surface-level behavior, but to infer the underlying intent and value structure guiding that behavior.

Maximum Entropy IRL [64] was among the first to formalize this framework probabilistically. Later works extended this to high-dimensional domains using deep neural networks, giving rise to Deep IRL [62] and Maximum Entropy Deep IRL (MEDIRL) [15]. These methods demonstrated the ability to learn socially compliant behavior such as path preference, crowd avoidance, and personal space preservation. Recent advances extend these ideas to real-world multi-agent pedestrian settings [6], where robot behavior is learned from unstructured crowd interactions.

However, most IRL-based approaches rely on a scalar reward function that conflates multiple social objectives - such as safety, efficiency, legibility, and predictability - into a single value. This limits interpretability and makes it difficult to prioritize different behaviors depending on context.

Limitations of Monolithic Learning Models

Despite their advantages, many learning-based methods suffer from limited interpretability and context sensitivity. Trajectory forecasting models provide predictions but do not explicitly encode communicative intent. IRL-based methods model behavior implicitly through a scalar reward, making it challenging to modulate robot motion between competing social priorities, such as clarity vs. conformity.

Moreover, learning-based methods often assume perfect knowledge of human intent and fail to account for ambiguity in interaction scenarios. For example, when the robot is unsure which direction a human intends to pass, relying solely on prediction may lead to hesitation or unsafe behavior.

These limitations motivate approaches like S-MEDIRL and our own vector-valued reward framework, which aim to make the trade-off between legibility and predictability explicit, interpretable, and learnable from data. Furthermore, deep IRL models rarely distinguish social communication goals, which limits their applicability in rich multi-agent contexts [11, 60].

2.4 Smooth Maximum Entropy Deep Inverse Reinforcement Learning (S-MEDIRL)

Smooth Maximum Entropy Deep Inverse Reinforcement Learning (S-MEDIRL) is a recent advancement in reward learning that builds on Maximum Entropy Deep IRL by incorporating spatial smoothing to improve generalization and robustness [29]. It addresses a key

limitation of deep IRL methods: the tendency to produce locally noisy and non-transferable reward fields, particularly in sparse or ambiguous environments.

The core idea of S-MEDIRL is to learn a dense reward map over the scene from expert demonstrations, while simultaneously enforcing spatial consistency via a bilateral smoothing loss. This regularization encourages the reward function to produce consistent values across spatially and semantically similar regions, allowing the robot to infer meaningful behavior in unobserved configurations.

S-MEDIRL operates in a discrete grid world, where the reward function is learned using a U-Net convolutional architecture and optimized using a combination of maximum entropy loss and bilateral filtering. Demonstrations are further augmented using extrapolated trajectories to improve robustness in narrow corridors and negotiation zones.

Evaluation in Habitat-Sim shows that S-MEDIRL significantly outperforms rule-based and standard IRL baselines in scenarios involving social negotiation, such as deciding when to yield or pass [29]. It achieves this without explicitly modeling social features - instead, legible and predictable behavior emerges from the learned reward structure.

Alternative formulations using graph attention and cooperative multi-agent learning are beginning to explore these distinctions more explicitly [11, 60].

While S-MEDIRL shows strong promise for socially compliant behavior, it models social qualities implicitly and as scalar values. It does not explicitly distinguish or control for different behavioral dimensions such as legibility and predictability, which limits its transparency and adaptability in real-world interactions.

2.5 Identified Research Gap

The literature surveyed highlights significant advances in both rule-based and learning-based approaches to social robot navigation. Rule-based systems like LPSNav [2] offer explicit control over legibility and predictability but rely on hand-engineered rules and lack flexibility. Learning-based models, particularly IRL methods [64, 15] and generative forecasting models [23, 49], provide strong generalization and adaptability but often treat social properties as implicit or monolithic objectives.

A critical limitation shared by most existing learning approaches is their use of a scalar reward function, which conflates multiple behavioral objectives into a single signal. This design makes it difficult to disentangle or prioritize different social criteria - such as whether

the robot should behave legibly to resolve ambiguity or predictably to conform to expectations. As a result, these systems cannot adapt their behavior in response to context-specific demands or human uncertainty.

Moreover, while frameworks like S-MEDIRL [29] implicitly model socially compliant behavior, they do not make the trade-off between legibility and predictability explicit. This observation aligns with recent efforts to formalize legibility as a learnable objective, separate from path optimality or goal-directedness [63]. This restricts the ability to reason about or modulate robot behavior in interaction-critical scenarios, such as head-on encounters or ambiguous lane-passing cases.

To address this gap, we propose a novel approach that integrates a vector-valued reward function, where legibility and predictability are treated as distinct but complementary objectives. By doing so, the robot can reason about its motion strategy in a multi-objective space and dynamically adapt its trajectory to better align with human perception and comfort. Additionally, recent guidelines stress the importance of evaluating social navigation behavior not only through geometric performance but also through human-centered metrics such as comfort, legibility, and clarity [17]. Our method builds on the strengths of S-MEDIRL but extends its formulation to support transparent and context-aware decision-making, with both simulation and real-world validation.

Chapter 3

Preliminaries

In this chapter, we formally define the navigation problem setting, the mathematical modeling assumptions, and the structure of the reward function used to guide socially-aware robot motion. We also present key notations and define legibility and predictability as measurable properties of a trajectory.

3.1 Problem Formulation

We consider the task of socially-aware robot navigation in dynamic, crowded indoor environments. The robot operates in close proximity to humans and must generate motion that not only avoids collisions, but also communicates its intent clearly and aligns with human expectations [38, 2].

At each time step, the robot receives sensory information from its onboard systems, including:

- A temporal history of its own motion (e.g., current and previous positions and velocities),
- A spatial representation of the static environment (e.g., an occupancy grid or semantic map),
- Observations of nearby humans, including their past or current positions and estimated motion,

- The robot’s current velocity or control state.

Based on this information, the robot must generate a trajectory that is not only safe and efficient but also socially compliant - that is, legible and predictable to nearby humans. The navigation policy must account for dynamic interactions, ambiguous intentions, and context-dependent social norms.

We formulate this as a sequential decision-making problem: given the current sensory observations, the robot must select a future trajectory that balances collision avoidance with socially-aware behavior. The details of how this information is encoded and how the navigation policy is derived are presented in Chapter 4.

3.2 Markov Decision Process Formulation

We formulate the socially-aware robot navigation task as a Markov Decision Process (MDP), defined by the tuple:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma), \quad (3.1)$$

where:

- $\mathcal{S}$ is the state space, which includes the robot’s current pose and velocity, the spatial context extracted from the environment encoder, and a countable set of motion features of surrounding humans (e.g., their positions and velocities),
- $\mathcal{A}$ is the continuous action space, parameterized by velocity commands (e.g., linear and angular velocities),
- $P(s'|s, a)$ is the state transition function, typically assumed deterministic under low-level motion controllers,
- $R(s)$ is the unknown reward function composed of interpretable components: legibility $L(s)$ and predictability $P(s)$,
- $\gamma \in [0, 1]$ is the discount factor controlling the planning horizon.

The robot seeks to learn a stochastic policy $\pi(a|s)$ that maximizes both cumulative reward and action entropy, following the principle of Maximum Entropy Reinforcement Learning [64]:

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^T R(s_t) + \alpha \cdot \mathcal{H}(\pi(\cdot|s_t)) \right], \quad (3.2)$$

where $\mathcal{H}(\pi(\cdot|s_t)) = -\sum_a \pi(a|s_t) \log \pi(a|s_t)$ is the entropy of the policy at state s_t , and α is a temperature parameter controlling the trade-off between reward maximization and exploration.

In our case, $R(s)$ is not provided explicitly but must be inferred from expert demonstrations $\mathcal{D} = \{\tau_1, \dots, \tau_N\}$, where each trajectory $\tau_i = (s_0, a_0, \dots, s_T, a_T)$ is generated by a socially aware expert (e.g., LPSNav planner [2]). This formulation naturally leads to the application of Inverse Reinforcement Learning (IRL), where the goal is to estimate the reward function $R_\theta(s)$ that explains the observed expert behavior.

Classical IRL methods model $R_\theta(s)$ as a scalar function [64, 15], but this work departs from that approach by proposing a vector-valued reward structure $R_\theta(s) = [L(s), P(s)]$, enabling the robot to reason about multiple social objectives in parallel. This setting is later optimized using the S-MEDIRL framework [29], extended here to support interpretable and context-sensitive planning.

3.3 Trajectory and Interaction Modeling

We define a trajectory ξ as a sequence of time-stamped robot positions over a planning horizon T :

$$\xi = \{x_1, x_2, \dots, x_T\}, \quad x_t \in \mathbb{R}^2. \quad (3.3)$$

Each trajectory implicitly reflects an *interaction goal* - a qualitative outcome in the context of human-robot interaction, such as “pass on the left,” “yield to oncoming agent,” or “merge behind a pedestrian.” These goals are not explicitly labeled, and the set of possible goals is not predefined; instead, they are treated as latent variables inferred from motion features and context.

At each time step t , the robot is in state s_t and has observed a partial trajectory $\xi_{1:t}$. To evaluate the social quality of its motion, we define a feature vector $\phi_t \in \mathbb{R}^d$ that aggregates relevant context from perception and motion encoders.

The feature vector is composed as follows:

$$\phi_t = [h_t \parallel e_t \parallel \hat{y}_t \parallel v_t^{\text{norm}}], \quad (3.4)$$

where $\parallel$ denotes vector concatenation, and:

- $h_t \in \mathbb{R}^{d_1}$ is the output of the Motion History Encoder, an LSTM that summarizes the robot’s recent trajectory,

- $e_t \in \mathbb{R}^{d_2}$ is the Environment Encoder’s CNN-based embedding of the local occupancy grid or semantic map,
- $\hat{y}_t \in \mathbb{R}^{d_3}$ represents predicted human expectations, such as anticipated path or interaction intent, extracted via attention or graph-based encoding [?],
- $v_t^{\text{norm}} \in \mathbb{R}^2$ is the normalized robot velocity vector.

This composite representation ϕ_t is fed into a vector-valued reward function $r_\theta(\phi_t) = [L_t, P_t]$, which outputs interpretable scores for legibility and predictability at each time step. These scores are then used to evaluate candidate trajectories ξ based on how clearly they signal the robot’s intent and how well they conform to expected social behavior [2, 29].

Importantly, by maintaining trajectory-level feature streams over time and modeling reward at each step, our framework supports both stepwise evaluation (for online planning) and trajectory-level aggregation (for batch IRL training). This allows learned behavior to generalize across diverse interaction settings while retaining interpretability of social signals at the motion level.

3.4 Legibility and Predictability Definitions

We adopt the formal definitions of *legibility* and *predictability* proposed in the probabilistic framework by Bastarache [2]. Let $\xi_{1:t}$ denote a partial observed trajectory of the robot up to time t , and let G be the set of possible interaction goals (e.g., pass on the left, yield, merge, overtake).

- **Legibility** refers to how well an observer can infer the robot’s intended interaction goal $g^* \in G$ based on the partial trajectory:

$$f_L(\xi_{1:t}) = \arg \max_{g \in G} P(g \mid \xi_{1:t}). \quad (3.4)$$

A trajectory is considered legible if $f_L(\xi_{1:t}) = g^*$ early in the interaction - i.e., the motion quickly and clearly communicates the robot’s intent.

- **Predictability** captures whether the robot’s motion is consistent with what a human would expect, given that the goal g^* is already known:

$$f_P(g^*) = \arg \max_{\xi} P(\xi \mid g^*). \quad (3.5)$$

That is, the trajectory should follow patterns that match the human observer’s expectations conditioned on the goal, such as smooth, direct paths or socially typical maneuvers.

These properties are not mutually exclusive, but often in tension. For example, an early deviation from a straight path may make the robot’s intent more legible (increasing $P(g^* \mid \xi_{1:t})$), but less predictable if it violates common motion conventions for the given goal (reducing $P(\xi \mid g^*)$). Our approach explicitly models this trade-off via a vector-valued reward function that quantifies both properties simultaneously.

3.5 Assumptions and Environment Setup

Our system is designed to operate in partially observable, dynamic indoor environments such as hallways, intersections, and shared spaces. The robot perceives its surroundings using a multi-modal sensing suite:

- A 2D LiDAR sensor is used to generate local occupancy grid maps, capturing static obstacles such as walls, furniture, and corners. The map is discretized into a grid centered on the robot’s current pose, providing input to the Environment Encoder [44].
- A monocular RGB camera is used to detect dynamic agents (i.e., humans) using a YOLOv8-based real-time object detector [12]. Detections are filtered and tracked across frames using a Kalman Filter or SORT-style tracker.
- Human motion is predicted using a trajectory forecasting model conditioned on recent positions, forming the input to the Human Expectation Encoder [1].
- The robot’s own motion history is tracked via onboard odometry or IMU, and encoded via an LSTM-based Motion History Encoder [37].

We assume that interaction goals (e.g., “pass on the left”, “yield”) are defined relative to the robot’s reference frame and not explicitly communicated by humans. All reasoning is performed under partial observability due to occlusions, dynamic changes, and limited sensor field-of-view.

The environment is continuously evolving, with uncertainty in human intent and motion. This setup motivates the need for interpretable, context-aware trajectory generation that dynamically balances legibility and predictability.

Chapter 4

Proposed Method

4.1 System Overview

The proposed navigation system enables a mobile robot to operate in dynamic, crowded indoor environments by generating trajectories that are both legible and predictable to nearby humans. The system integrates multi-modal perception, deep representation learning, and reward-based planning into a unified framework that dynamically adapts to social context and interaction uncertainty.

At the core of the method is a learned reward function $r_\theta(\phi_t)$, which evaluates the robot’s state based on extracted feature embeddings. These features are derived from recent robot motion, static environmental layout, human presence, and estimated human behavior. The reward outputs - representing legibility and predictability - are then used within the S-MEDIRL framework [29] to guide motion planning.

As illustrated in Figure 4.1, the system consists of six main components:

1. **Motion History Encoder (LSTM)** encodes the robot’s recent trajectory into a temporal embedding h_t that captures motion dynamics such as heading, speed variation, and turning behavior [37].
2. **Environment Encoder (CNN)** extracts a spatial representation e_t from a 2D occupancy grid map generated via LiDAR. This grid encodes static obstacles and free space [44].

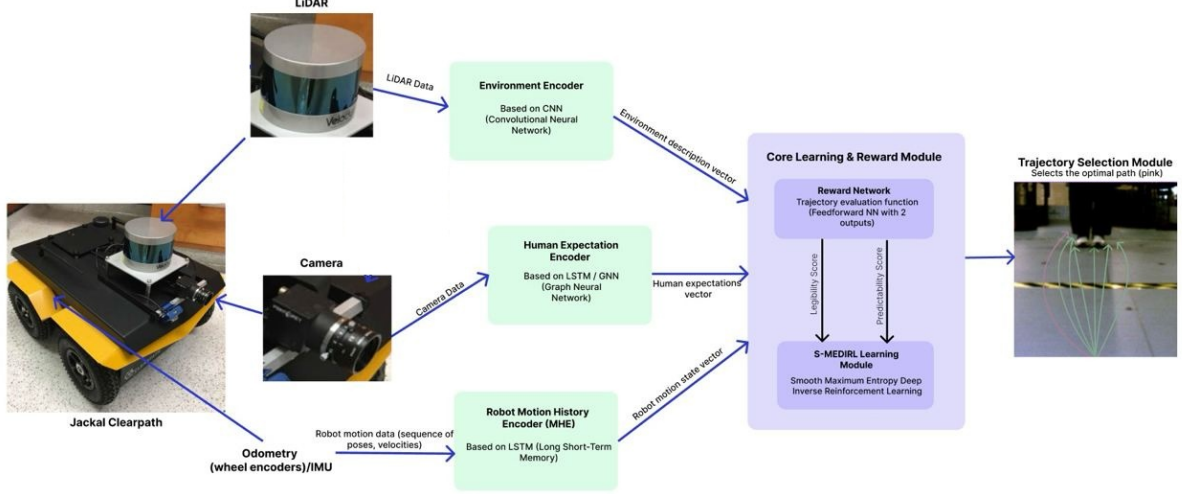


Figure 4.1: Overview of the proposed system architecture for socially-aware navigation.

3. **YOLOv8-Based Person Detection Module** processes RGB images from a monocular camera to detect and localize nearby humans in real time [12]. Bounding boxes are tracked using a Kalman filter to produce motion traces.
4. **Human Expectation Encoder** receives the detected human trajectories and occupancy context and outputs a feature vector $\hat{y}_t$ that encodes likely human responses or expectations [1, 58].
5. **Velocity Normalization Module** transforms the robot’s linear and angular velocity into a scale-invariant form v_t^{norm} for generalization across speeds and environments.
6. **Reward Network** concatenates all features into a single vector:

$$\phi_t = [h_t \parallel e_t \parallel \hat{y}_t \parallel v_t^{\text{norm}}] \quad (4.1)$$

and outputs two scores: legibility L_t and predictability P_t . These jointly form a vector-valued reward signal.

The reward function is trained on expert trajectories produced by LPSNav [2], a rule-based planner that encodes socially compliant behaviors such as yielding and early intent signaling. Unlike imitation learning, we do not copy behavior but distill the emergent legibility/predictability trade-offs into a differentiable reward structure.

At inference time, the robot processes sensor data from LiDAR and camera in real time, encodes its current situation, and uses the learned reward to evaluate candidate motion trajectories. The S-MEDIRL framework optimizes over these trajectories to select one that maximizes social interpretability and behavioral alignment with human expectations [64, 29]. This enables legible and predictable motion that adapts fluidly to changing social contexts and navigation goals.

4.2 Motion History Encoder (MHE) with LSTM Model

In this section, we describe the design and training of the Motion History Encoder (MHE), a neural module based on Long Short-Term Memory (LSTM) architecture [26]. The MHE learns temporal patterns in the robot’s past motion to predict short-term future movement. This predictive capacity is essential for estimating dynamic legibility and context-aware planning in our navigation system.

The MHE outputs an embedding h_t that encodes recent kinematic behavior (e.g., turning, acceleration) into a latent feature vector. This embedding is used downstream in the reward computation module as part of the feature vector ϕ_t .

4.2.1 Data Preparation and Preprocessing

The input to the MHE consists of time-series sequences of robot states extracted from the *nuScenes* dataset [3]. This dataset contains detailed annotations of autonomous vehicle motion in urban traffic scenarios, including position, velocity, and timestamped sensor data.

Each robot state is defined as a vector:

$$s_t = [x_t, y_t, v_t] \in \mathbb{R}^3 \quad (4.2)$$

representing the robot’s 2D position (x_t, y_t) and linear velocity v_t at time t . All trajectories are sampled at regular intervals of $\Delta t = 0.5s$.

Preprocessing steps include:

- *Outlier filtering*: Removing sequences with missing timestamps, GPS dropouts, or excessive jump discontinuities;

- *Min-Max normalization*: Scaling each s_t to $[0, 1]$ per scene, improving learning stability and convergence;
- *Segmentation*: Sliding window extraction of training pairs. Each input sequence of 5 historical states is mapped to the next future state:

$$\{s_{t-4}, s_{t-3}, \dots, s_t\} \rightarrow s_{t+1}. \quad (4.3)$$

This transforms raw trajectory data into a supervised learning problem with fixed-length sequences. The window size (5 past steps) was selected based on prior work in short-term trajectory forecasting [1, 37], balancing model reactivity with temporal context.

Figure 4.2 illustrates sample training trajectories from the nuScenes dataset. The diversity in path shapes, speeds, and turns ensures the model generalizes across various navigation styles and environments.

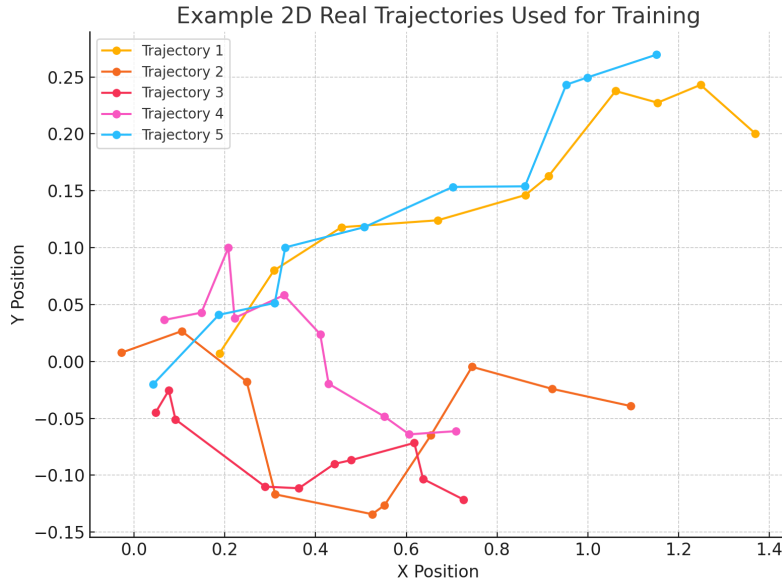


Figure 4.2: Examples of 2D robot trajectories used during training. Each curve represents a sequence of (x_t, y_t, v_t) tuples from real navigation episodes in nuScenes.

4.2.2 LSTM Model Architecture

The Motion History Encoder (MHE) is implemented as a sequence-to-one regression model using a two-layer Long Short-Term Memory (LSTM) architecture [26]. This module learns

to predict the robot’s immediate future state based on a sequence of its recent motion. The resulting embedding serves as a temporally-informed representation h_t , which contributes directly to the reward evaluation for legibility and predictability.

Input and Output Structure

The input to the model is a normalized sequence of 5 consecutive robot states:

$$\mathcal{X} = \{s_{t-4}, s_{t-3}, \dots, s_t\}, \quad s_i = [x_i, y_i, v_i] \in \mathbb{R}^3 \quad (4.4)$$

structured as a tensor of shape $(5, 3)$, where the first dimension corresponds to the 5 consecutive time steps and the second dimension corresponds to the 3 state features (x, y, v) . The prediction target is the next state:

$$\hat{s}_{t+1} = [x_{t+1}, y_{t+1}, v_{t+1}] \in \mathbb{R}^3. \quad (4.5)$$

This format aligns with standard practices in trajectory forecasting, as seen in works such as Social LSTM [1] and Trajectron++ [49], where past motion is encoded to anticipate future dynamics in interactive environments.

Model Architecture

The architecture is defined as follows:

- **LSTM Layer 1:** 64 hidden units, ‘return_sequences=True’
- **LSTM Layer 2:** 32 hidden units, ‘return_sequences=False’
- **Dropout:** 0.2 dropout rate to prevent overfitting
- **Dense Output Layer:** 3 neurons with linear activation to predict $[x, y, v]$

This stacked LSTM configuration captures both short-term kinematic trends and higher-order temporal structure, as supported by prior work in socially-aware navigation [37, 14]. The shallow design ensures computational efficiency while remaining expressive for short-horizon prediction tasks.

Training Setup

The network is trained using the Mean Squared Error (MSE) loss:

$$\mathcal{L}_{\text{MSE}}(\theta) = \frac{1}{3} \sum_{i=1}^3 (s_{t+1}^i - \hat{s}_{t+1}^i)^2, \quad (4.6)$$

with the Adam optimizer and a learning rate of 1×10^{-3} . Training is performed over 50 epochs with a batch size of 64. Early stopping is employed based on validation set performance to prevent overfitting.

Output Usage

At inference time, the MHE produces a 32-dimensional hidden state vector from the second LSTM layer, which is passed forward as the feature embedding h_t . This embedding summarizes the robot’s recent motion profile and contributes to the full state representation ϕ_t used in reward computation.

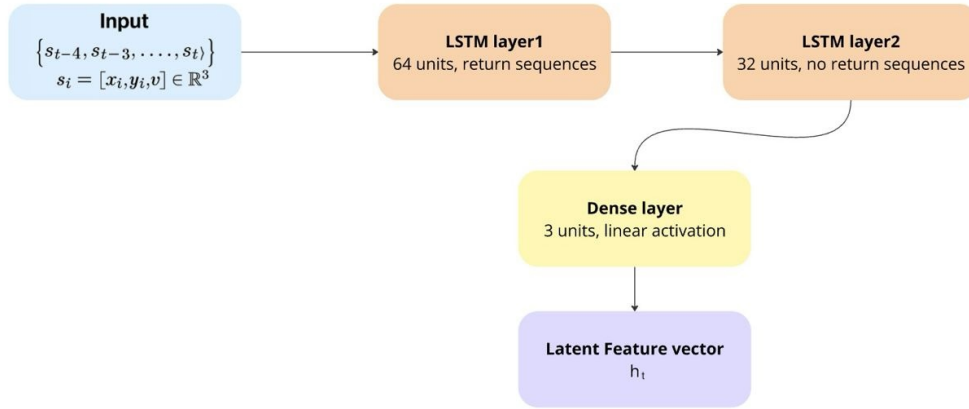


Figure 4.3: Motion History Encoder (MHE) architecture. The input is a sequence of 5 recent robot states $s_i = [x_i, y_i, v_i] \in \mathbb{R}^3$, which passes through two stacked LSTM layers followed by a dense output layer. The final output is a latent feature vector h_t encoding recent motion dynamics.

By integrating the MHE into the end-to-end reward learning pipeline, the robot is able to reason about how recent motion affects social interpretability - improving its ability to signal intent and align with human expectations.

4.2.3 Model Training

The Motion History Encoder (MHE) model is trained on the preprocessed trajectory dataset using a supervised learning approach. The objective is to minimize the Mean Squared Error (MSE) between the predicted and actual future robot states:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^N \|\hat{s}_{t+1}^{(i)} - s_{t+1}^{(i)}\|_2^2, \quad (4.7)$$

where N is the batch size and $s_{t+1}^{(i)}$ is the ground truth state in sample i .

Training Procedure

We use the Adam optimizer with an initial learning rate of 1×10^{-3} and apply weight decay ($\lambda = 10^{-5}$) for regularization. The model is trained for up to 100 epochs using early stopping with a patience of 10 epochs, monitoring the validation loss. The dataset is split as follows:

- 80% training set,
- 10% validation set,
- 10% test set.

A small batch size of 4 is used due to GPU memory constraints and the relatively small size of each sequence sample. Dropout with a rate of 0.2 is applied between LSTM layers to reduce overfitting.

Evaluation Metrics

In addition to MSE, we track the following motion prediction metrics commonly used in trajectory forecasting literature [1, 49]:

- **Average Displacement Error (ADE)**: mean Euclidean distance (in meters) between predicted and ground truth over the whole trajectory,
- **Final Displacement Error (FDE)**: Euclidean distance (in meters) between the predicted final point and the ground truth final point.

These metrics are computed on the test set and reported in Chapter 5.

Training Dynamics

Figure 4.4 shows the training loss curve across epochs. The steady decline in MSE suggests that the model successfully learns motion patterns from the data. No signs of overfitting are observed, as the validation loss follows a similar decreasing trend. This indicates the model generalizes well to unseen trajectories.

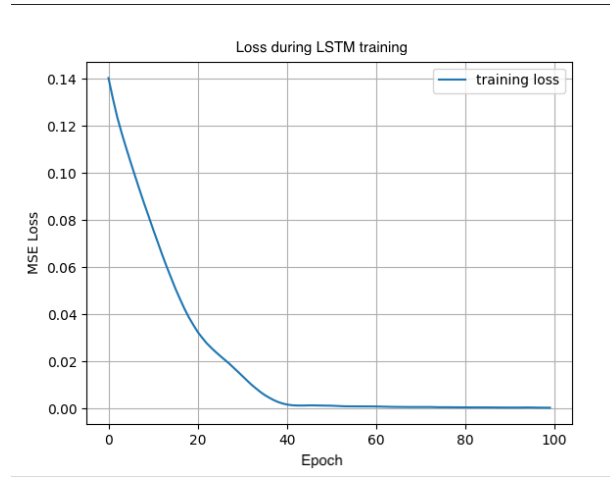


Figure 4.4: Training loss curve of the LSTM model over 100 epochs. The decreasing MSE loss indicates consistent learning.

4.2.4 Evaluation and Results

To evaluate the performance of the Motion History Encoder (MHE), we tested the trained model on a held-out set of robot trajectories that were not used during training or validation. The goal was to assess the model’s ability to generalize and accurately forecast short-term motion based on past experience.

Quantitative Metrics

The following metrics were computed on the test set:

- **Mean Squared Error (MSE):** 0.000718 (normalized coordinates)
- **Average Displacement Error (ADE):** 0.0229

- **Final Displacement Error (FDE):** 0.0341

These metrics are widely adopted in human trajectory forecasting [1, 49] and offer a reliable measure of how well the predicted paths align with the ground truth. The small values for ADE and FDE indicate strong agreement over both the full prediction horizon and the final position.

As an example, for a single test sample:

- **Ground Truth:** [0.3296, 0.7399, 0.6938]
- **Predicted:** [0.3191, 0.7189, 0.7082]

Qualitative Results

In addition to numerical evaluation, we conducted visual comparisons between predicted and actual trajectories. Figure 4.9 shows four representative examples, demonstrating the model’s generalization ability:

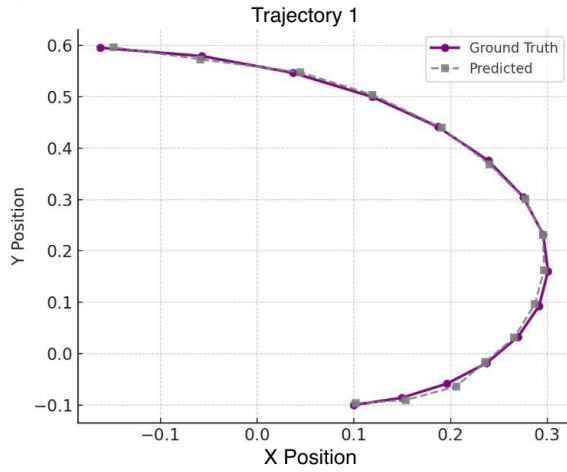


Figure 4.5: Smooth arc-shaped movement: the prediction closely follows a curved path with minimal deviation.

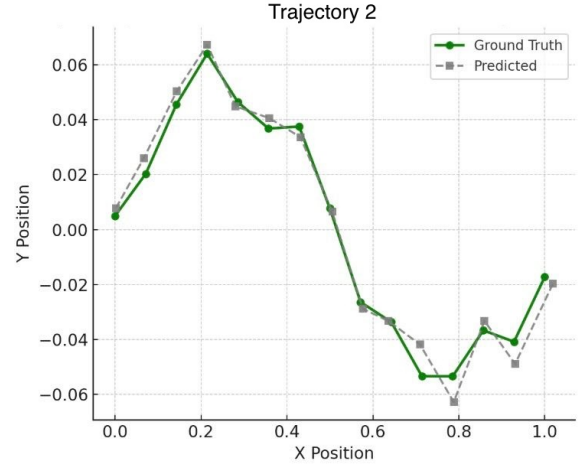


Figure 4.6: Motion with local turns and variations: the model adapts well to non-linear, human-like behavior.

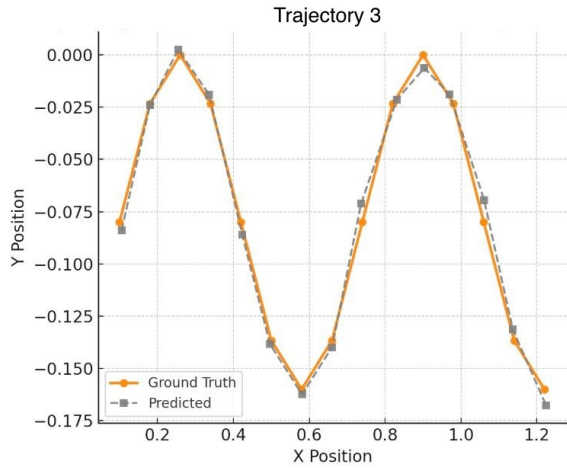


Figure 4.7: Oscillating sine-like motion: the prediction accurately captures the periodic structure.

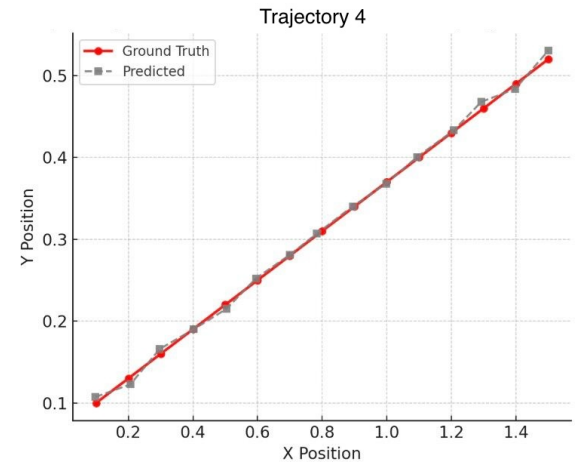


Figure 4.8: Linear diagonal movement: almost perfect prediction with overlap throughout the path.

Figure 4.9: Comparison of actual and predicted robot trajectories across different motion types. The model generalizes well to both structured and irregular movement patterns.

These visualizations confirm that the LSTM-based encoder captures both low-level kinematic changes and higher-order spatial structures, even in non-trivial motion scenarios. The ability to handle non-linear and oscillatory patterns is essential for real-world navigation in complex environments.

Error Analysis

Slight deviations were observed in sharply curved or stop-and-go trajectories, where uncertainty accumulates rapidly. Such behaviors are inherently harder to predict due to sudden velocity changes and sensor noise, as noted in prior work [39].

Nevertheless, the MHE demonstrates robust generalization, and its prediction quality is sufficient for reliable downstream planning and reward evaluation.

4.2.5 Discussion of MHE Contribution

The Motion History Encoder (MHE) provides a compact and interpretable embedding of recent robot motion, enabling short-term prediction in dynamic environments. Its use of stacked LSTM layers allows the system to retain temporal dependencies without overfitting to surface-level motion patterns [1].

This module contributes directly to the feature vector ϕ_t used in reward computation, capturing how the robot’s recent behavior influences its perceived intent. Similar approaches have shown strong potential in social navigation pipelines [37], where short-horizon forecasting improves safety, coordination, and legibility.

By integrating the MHE into the learning-based reward architecture, our system gains the ability to reason about motion consistency and continuity - critical for socially compliant and predictable robot behavior.

4.3 Environment Encoder (CNN)

A central element of our robot navigation system is the **Environment Encoder**, a Convolutional Neural Network (CNN) designed to extract meaningful spatial features from LiDAR-based representations of the environment. This encoder plays a critical role in enabling the robot to understand where obstacles are, identify traversable areas, and plan safe paths in real time.

4.3.1 Purpose and Context

The robot operates in cluttered indoor environments such as hallways, rooms, and intersections, where accurate perception of static structures is essential for safe and socially compliant navigation. While dynamic agents (i.e., humans) are detected via monocular camera and YOLOv8-based real-time object detection, static elements - including walls, furniture, and equipment - are sensed using a planar 2D LiDAR scanner.

LiDAR readings are accumulated and converted into a local *Occupancy Grid Map* (OGM), a discretized 2D matrix representation of the surrounding environment:

$$M \in \mathbb{R}^{H \times W}, \quad M_{ij} \in \{-1, 0, 1\}, \quad (4.8)$$

where each cell represents unknown space (-1), free space (0), or occupied space (1). The grid is centered on the robot and spans a fixed area around it (e.g., $8 \text{ m} \times 8 \text{ m}$), with a typical resolution of 0.1 m per cell, yielding an input size of 80×80 pixels.

This occupancy grid is interpreted as a single-channel grayscale image and passed to the **Environment Encoder**, which is implemented as a convolutional neural network (CNN). The encoder extracts high-level spatial features such as:

- wall proximity and alignment,
- presence of bottlenecks or corridors,
- obstacle density and distribution.

The CNN compresses this information into a compact embedding e_t that encodes the current spatial configuration around the robot. This encoding is critical for downstream modules - particularly the reward function and Human Expectation Encoder - which must reason about motion feasibility, visibility, and navigability.

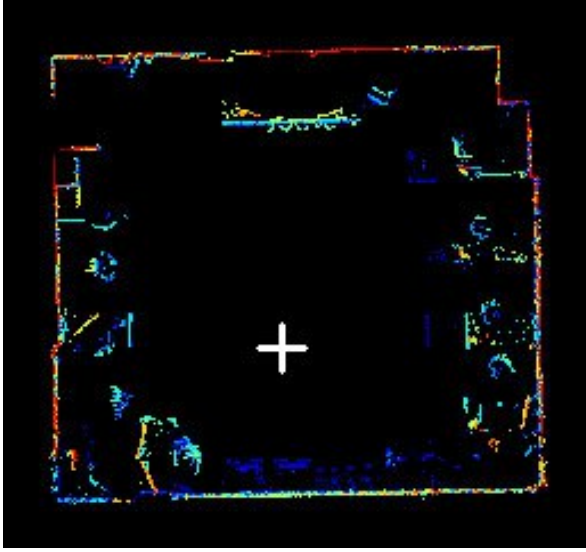
Such CNN-based perception has proven effective in end-to-end navigation pipelines [44, 31], and allows the system to adaptively respond to environmental complexity without requiring hand-coded spatial heuristics.

4.3.2 Input Data: Occupancy Grid from LiDAR

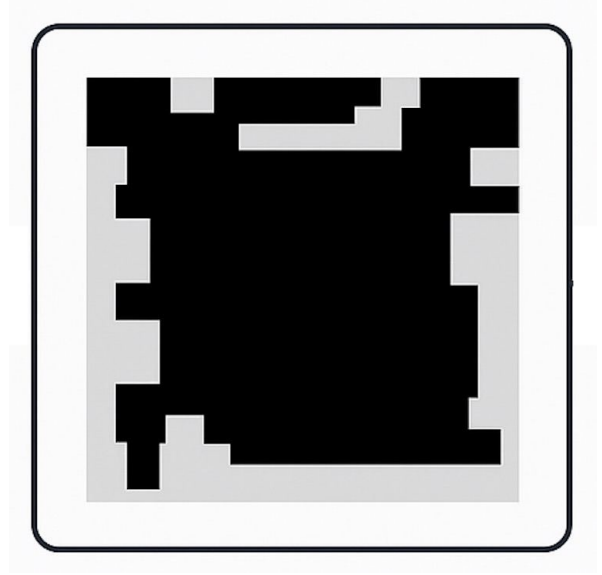
The LiDAR data is first converted into an `OccupancyGrid` using SLAM (Simultaneous Localization and Mapping) tools such as Cartographer. Each grid cell stores the probability of being occupied:

$$I(x, y) = \mathbb{P}(\text{occupied} \mid \text{LiDAR data}). \quad (4.9)$$

This results in a 2D matrix $I \in \mathbb{R}^{H \times W}$, where higher intensity values indicate obstacles (bright areas) and lower values correspond to free space (dark regions).



(a) Raw LiDAR intensity map



(b) Occupancy grid representation for CNN input

Figure 4.10: Transformation from raw LiDAR data to occupancy grid: The raw LiDAR scan (a) is processed using SLAM (e.g., Cartographer) to produce an occupancy grid (b), where white cells indicate high obstacle probability. This processed map is used as input to the Environment Encoder CNN.

4.3.3 CNN Architecture and Image Processing Pipeline

To extract compact and informative spatial features from LiDAR-based occupancy maps, we implement a lightweight convolutional neural network (CNN) encoder. The input to the encoder is a 64×64 grayscale image, derived from the occupancy grid and normalized to the $[0, 1]$ range. Each pixel encodes the local occupancy state: free (0), occupied (1), or unknown (0.5).

The CNN architecture consists of the following layers:

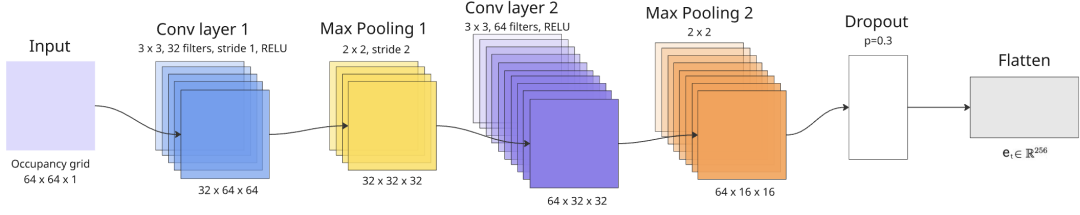


Figure 4.11: Architecture of the CNN-based Environment Encoder. The input is a 64×64 occupancy grid derived from LiDAR data. The network consists of two convolutional layers (with ReLU activation), interleaved with 2×2 max pooling layers, followed by dropout ($p = 0.3$) and a flattening layer. The final output is a 256-dimensional spatial feature vector $e_t \in \mathbb{R}^{256}$, which encodes the environmental layout for downstream reasoning and planning.

- **Conv Layer 1:** 32 filters, 3×3 kernel, stride 1, ReLU activation.
- **Max Pooling:** 2×2 pool size, stride 2.
- **Conv Layer 2:** 64 filters, 3×3 kernel, ReLU activation.
- **Max Pooling:** 2×2 pool size.
- **Flatten Layer:** Outputs a vector $e_t \in \mathbb{R}^{256}$.

Dropout with a rate of 0.3 is applied after the second convolutional layer to reduce overfitting. All convolutional layers use zero-padding to preserve spatial dimensions prior to pooling.

The resulting feature vector e_t captures global spatial layout - such as corridor width, obstacle density, and bottleneck geometry - while remaining computationally efficient. These features are passed downstream to the reward function and human expectation modules.

Unlike CNNs used in image classification, our encoder is not trained to assign labels. Instead, it acts as a differentiable spatial reasoning module that learns to produce embeddings useful for downstream decision-making. All CNN weights are updated end-to-end during reward function optimization via inverse reinforcement learning.

The use of CNNs for occupancy encoding is well-established in navigation literature [44, 31, 49], and allows the robot to learn generalizable planning strategies without relying on handcrafted features or explicit map interpretation.

4.3.4 Training Methodology

To train the Environment Encoder, we used real-world occupancy grid representations derived from the **nuScenes** dataset [3], which offers diverse multimodal recordings from urban driving environments. Specifically, we extracted 2D top-down occupancy grids from 360° LiDAR point clouds by projecting points onto the ground plane and discretizing the space into fixed-resolution cells.

Each cell of the grid was labeled according to:

$$I(x, y) = \begin{cases} 1 & \text{if any LiDAR return is registered in the cell,} \\ 0 & \text{otherwise.} \end{cases} \quad (4.10)$$

These binary maps were resized to 64×64 and normalized to the $[0, 1]$ range. Each resulting grid was then used as both the input and target for training the CNN, forming a **self-reconstruction task** that forces the encoder to learn spatial priors and obstacle geometry [5].

Supervision Signal The CNN is trained in a supervised manner using Binary Cross Entropy (BCE) loss:

$$\mathcal{L}_{BCE} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]. \quad (4.11)$$

This loss encourages the model to correctly classify each grid cell as occupied or free. While no semantic labels are used, the network learns to extract spatial structure, such as walls, corridors, and clustered obstacles, which are essential for downstream trajectory planning [44, 31].

Training Configuration

- Optimizer: Adam
- Learning Rate: 1×10^{-4}
- Batch Size: 16
- Epochs: 50

We split the data into 80% training and 20% validation scenes to ensure generalization across diverse spatial configurations. The model converged within the first 15 epochs, with validation accuracy exceeding 90% based on pixel-wise classification. In practice, the network consistently produced interpretable embeddings across different map topologies, from narrow passages to open intersections.

These results demonstrate that even without semantic supervision, the CNN learns a compact representation of spatial occupancy that is robust to occlusions, sparsity, and structural variability - making it well-suited for integration into learned reward functions and interaction-aware navigation systems.

4.3.5 Qualitative Results and Output Interpretation

After training, we compare predicted occupancy maps to ground truth maps. Figure 4.12 shows side-by-side visualizations of:

- Ground truth occupancy grid (SLAM-based).
- Predicted occupancy grid generated by CNN from LiDAR-derived inputs.

Intermediate CNN feature maps reveal detection of obstacles, corridor width, and open areas, highlighting how the model internalizes spatial affordances.

4.3.6 Contribution to Navigation

The output embedding e_t from the Environment Encoder is concatenated with temporal and dynamic features from the Motion History Encoder (h_t), forming the full state vector:

$$\phi_t = [h_t \parallel e_t \parallel \hat{y}_t \parallel v_t^{\text{norm}}]. \quad (4.12)$$

This unified representation enables the robot to:

- Detect and avoid static obstacles, walls, and narrow gaps,
- Exploit geometric cues from the scene layout (e.g., corridor alignment, obstacle density),

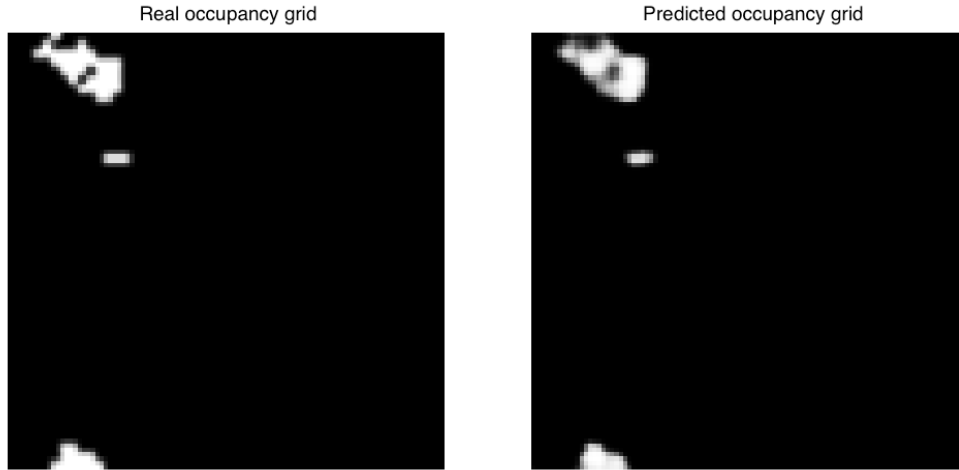


Figure 4.12: Left: Real occupancy grid from SLAM. Right: Predicted grid from CNN output.

- Adapt its motion policy to structural constraints in the environment,
- Plan socially compliant trajectories even in previously unseen indoor spaces.

By learning a latent representation of occupancy maps, the CNN eliminates the need for hand-engineered spatial rules (e.g., “keep fixed distance from walls”) and replaces them with data-driven feature extraction, as demonstrated in prior work on spatial encoders [49].

4.3.7 Discussion of Environment Encoder Contribution

The Environment Encoder bridges low-level perception and high-level trajectory planning by transforming sparse, noisy LiDAR readings into a dense spatial embedding suitable for learning-based decision making. When combined with the temporal modeling of the Motion History Encoder, it equips the robot with both scene awareness and motion context.

This synergy allows the reward function to reason jointly about spatial feasibility and motion clarity - two critical components for generating legible and predictable navigation in real-world human environments.

4.4 Human Trajectory Forecasting Using LSTM and CVAE

To enable safe and socially compliant robot navigation, it is essential to accurately anticipate the future motion of nearby pedestrians. This section presents a dedicated trajectory prediction framework for human agents, inspired by Trajectron++ [49], but redesigned to suit real-time robotic applications. Our approach combines real-world human detection (via YOLOv8 [12]), motion tracking (via Kalman filtering), temporal modeling (via LSTM), and multimodal probabilistic forecasting (via CVAE).

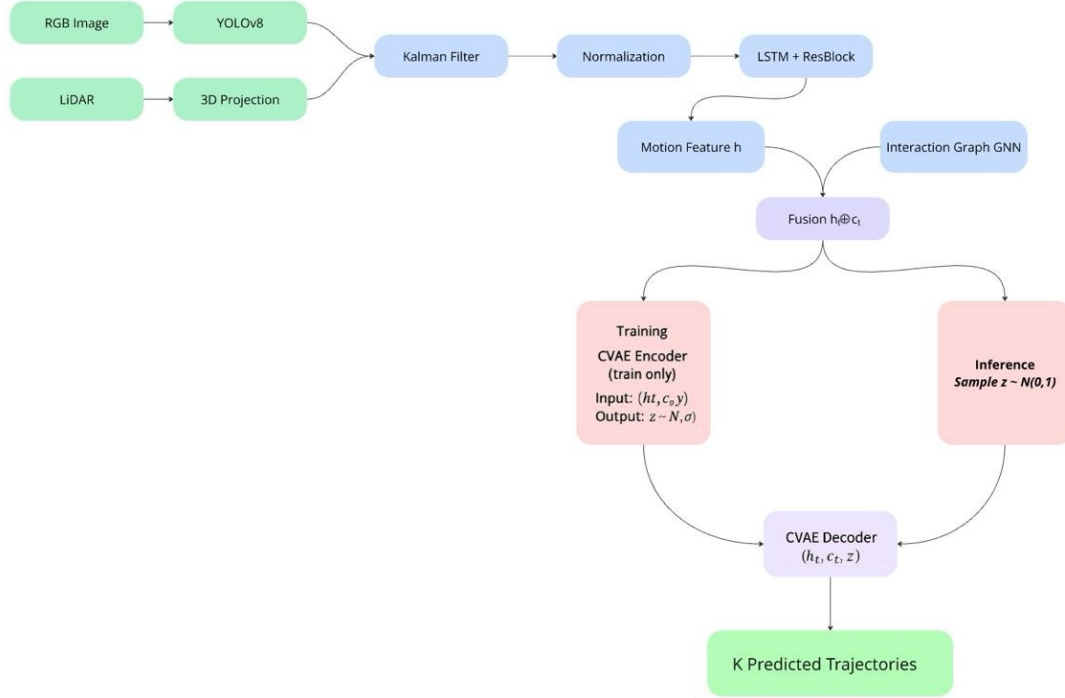


Figure 4.13: Overall architecture of the Human Encoder. The pipeline integrates RGB- and LiDAR-based perception, Kalman filtering, temporal encoding (LSTM + ResBlock), interaction modeling via a GNN, and probabilistic trajectory generation using a CVAE.

4.4.1 Input Modalities and Preprocessing

Pedestrian detection is performed using the YOLOv8 object detection model [12] applied to RGB frames captured by the robot’s monocular front-facing camera. For each detection, the centroid of the bounding box is projected into 3D world coordinates (x, y, z) using known extrinsic camera-LiDAR calibration and height-based LiDAR slicing. This allows 2D image-based detections to be grounded in metric space.

Figures 4.14 and 4.15 illustrate this pipeline: YOLOv8 [12] produces person detections in image space, which are then fused with occupancy information to estimate global position.



(a) Raw RGB camera input



(b) YOLOv8 person detection

Figure 4.14: Camera-based pedestrian perception. (a) Raw RGB camera view from the robot without annotation. (b) YOLOv8 [12] detection showing persons with bounding boxes and confidence scores.

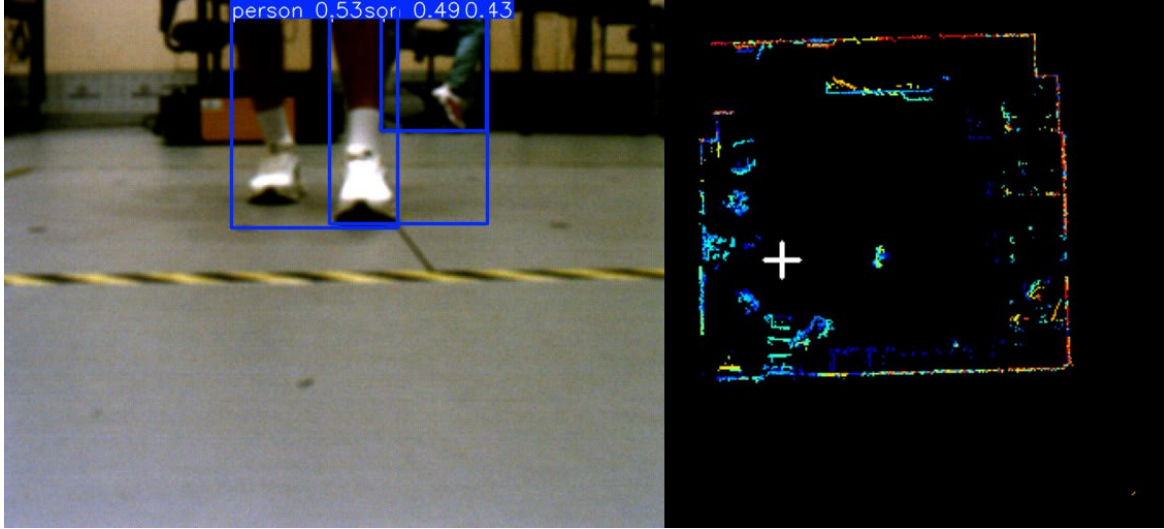


Figure 4.15: Fusion of RGB and LiDAR data for spatial localization. Left: YOLOv8 detections [12]. Right: corresponding LiDAR occupancy grid used to project 2D detections into 3D coordinates.

To maintain temporal continuity and smooth out sensor noise, we apply a linear Kalman Filter [61] to each detected person. The Kalman Filter tracks the centroid position over time and predicts the next location if detection is lost momentarily, improving robustness in occluded scenes.

The result is a time-aligned sequence of smoothed 3D coordinates for each agent:

$$\{(x_t, y_t, z_t)\}_{t=t_0}^{t_0+T-1}. \quad (4.13)$$

Note that z_t corresponds to the height coordinate from the dataset; for ground robots this value is approximately constant and does not influence the trajectory prediction. Each agent’s trajectory is centered and normalized with respect to the robot’s local coordinate frame. Sequences are then resampled using a sliding window approach:

- `seq_len` = 5 historical steps for input,
- `pred_len` = 8 future steps as supervision (for supervised training or prediction).

This windowing ensures consistent input length for downstream models such as the Human Expectation Encoder. Sequence length was selected based on empirical validation and

to match the timescale of human-reactive navigation (≈ 1.3 –2 seconds at typical walking speed).

Bounding box association over time is performed using frame-to-frame centroid distance and Intersection-over-Union (IoU) thresholds. If a match is not found for several consecutive frames, the track is terminated.

4.4.2 LSTM-Based Human Encoder

To model the temporal dynamics of human motion, each agent’s trajectory history is processed using a recurrent neural network based on Long Short-Term Memory (LSTM) units [26]. The input to the encoder consists of a fixed-length sequence of 3D coordinates:

$$\mathbf{x}_t = [(x_{t-\text{seq_len}}, y_{t-\text{seq_len}}, z_{t-\text{seq_len}}), \dots, (x_t, y_t, z_t)] \in \mathbb{R}^{\text{seq_len} \times 3}. \quad (4.14)$$

This sequence is first normalized relative to the robot’s current position and velocity, ensuring spatial consistency across trajectories.

The LSTM encoder architecture comprises the following components:

- **LSTM layer:** A single-layer LSTM with `hidden_dim` = 128 units processes the input sequence and generates hidden states $h_1, h_2, \dots, h_t$. LSTMs have been widely adopted for human motion modeling [1], as they capture long-range temporal dependencies in sequential data. We also experimented with bidirectional LSTMs [21] to capture both past and future context within the observed sequence.
- **Residual block (ResBlock):** To improve gradient flow and stabilize training, a residual connection is added around the LSTM output. This helps preserve low-level temporal features and enables the network to learn both incremental and global motion patterns. Residual LSTM-style architectures have shown benefits in various temporal domains [33].
- **Dense layer:** The final hidden state is passed through a fully connected (dense) layer to produce a compact latent embedding $\mathbf{h}_t \in \mathbb{R}^d$ that encodes the motion intent and kinematic signature of the agent.
- **Regularization:** A dropout layer ($p = 0.2$) is used after the dense layer to prevent overfitting, especially in low-data regimes.

Formally, the final encoding is given by:

$$\mathbf{h}_t = \text{Dense}(\text{ResBlock}(\text{LSTM}(\mathbf{x}_t))). \quad (4.15)$$

This latent vector $\mathbf{h}_t$ is then forwarded to the CVAE decoder for trajectory forecasting [49]. It captures not only the direction and magnitude of movement, but also subtle temporal features like pauses, speed variations, and turns.

Empirical experiments showed that using residual connections improved performance on non-linear and stop-go trajectories. Bidirectional LSTMs further enhanced the encoder’s ability to generalize to symmetric motions (e.g., switching between left/right branches in intersections), though at the cost of increased latency. In the final implementation, we use a unidirectional LSTM for real-time efficiency.

4.4.3 Modeling Inter-Agent Interaction

To enable socially aware trajectory forecasting, it is essential to account for how pedestrians influence each other’s movement. Our framework models these interactions using a time-varying graph structure, capturing both spatial proximity and temporal evolution between agents [49, 58, 27].

Graph Construction. At each time step t , a fully dynamic interaction graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ is constructed, where:

- $\mathcal{V}_t$ denotes the set of all pedestrians detected at time t ,
- $\mathcal{E}_t$ contains undirected edges (i, j) between agents i and j if their Euclidean distance is below a threshold radius R :

$$\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\| < R. \quad (4.16)$$

This graph captures the set of nearby agents whose motion is likely to affect each other and evolves dynamically as new agents enter or exit the scene [23].

Edge Encoding. To capture how proximity changes over time, each edge (i, j) is associated with a temporal sequence of pairwise distances:

$$\Delta_{ij}(t) = [\|\mathbf{x}_i(t - k) - \mathbf{x}_j(t - k)\|]_{k=0}^{\text{seq_len}-1}. \quad (4.17)$$

This sequence is passed through a lightweight LSTM encoder to produce an edge embedding $\mathbf{e}_{ij} \in \mathbb{R}^d$, which encodes temporal trends such as approaching, moving away, or walking in parallel [1].

Context Aggregation via Attention. Each agent i aggregates encoded information from its neighbors $j \in \mathcal{N}(i)$ using a soft attention mechanism. This allows the model to weigh nearby agents based on their relevance to the target’s future motion. The attention-weighted context vector is computed as:

$$\mathbf{c}_i = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \cdot \mathbf{h}_j, \quad (4.18)$$

$$\alpha_{ij} = \frac{\exp(\text{score}(\mathbf{h}_i, \mathbf{h}_j))}{\sum_{k \in \mathcal{N}(i)} \exp(\text{score}(\mathbf{h}_i, \mathbf{h}_k))}, \quad (4.19)$$

where $\mathbf{h}_j$ is the motion encoding of agent j , and $\text{score}(\cdot)$ is a learned compatibility function (e.g., scaled dot-product or MLP), similar to that used in graph attention networks [57].

This attention mechanism enables each agent to focus on the most influential neighbors (e.g., those on a potential collision path), while down-weighting distant or irrelevant ones.

Final Representation. The concatenated vector $\mathbf{z}_i = \text{concat}(\mathbf{h}_i, \mathbf{c}_i)$ is used as the conditioning input for the CVAE decoder during training and inference.

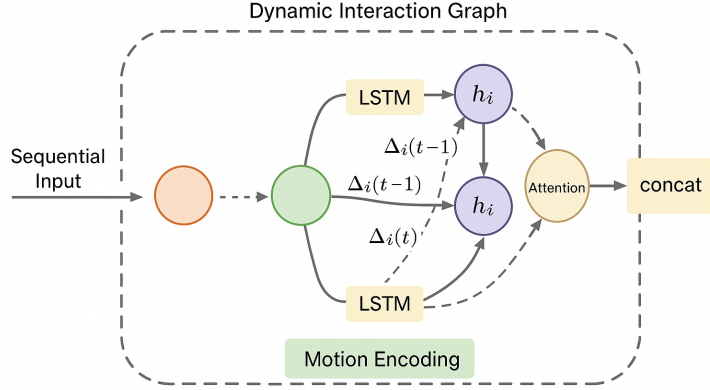


Figure 4.16: Inter-agent interaction graph. Each node represents a pedestrian. Edges encode time-series proximity features between neighboring agents. These are processed by LSTM encoders, and attention-based aggregation produces a socially informed context vector used for future trajectory forecasting.

By explicitly modeling interactions as a graph, this module enables the system to reason about complex crowd behavior, such as coordinated walking, yielding at intersections, or avoiding collisions - all of which are critical for human-aware navigation [47].

4.4.4 Multimodal Future Prediction via CVAE

Human motion is inherently stochastic and multimodal: at any given time, a pedestrian may choose among multiple plausible future actions (e.g., go left, continue straight, stop). To address this uncertainty, we employ a **Conditional Variational Autoencoder (CVAE)** framework [36] to generate diverse and probabilistic predictions of future human trajectories.

Motivation. Unlike deterministic models that produce a single best-guess trajectory, the CVAE enables the generation of multiple hypotheses, allowing the robot to reason about possible future outcomes and plan accordingly. This is especially important in crowded or ambiguous scenes [39, 49].

Architecture Overview. The CVAE consists of three main components [42]:

- **Encoder:** Takes the motion history and the ground-truth future trajectory and maps them to a latent posterior distribution $q(z|x, y)$.
- **Latent Space:** A multivariate Gaussian distribution $\mathcal{N}(\mu, \sigma)$ from which samples z are drawn.
- **Decoder:** Receives the sampled z , along with the encoded motion history and social context, and generates predicted future trajectories.

Training Phase. During training, the CVAE learns to minimize a combined loss:

$$\mathcal{L}_{\text{CVAE}} = \mathcal{L}_{\text{recon}} + \beta \cdot \mathcal{L}_{\text{KL}}, \quad (4.20)$$

where:

- $\mathcal{L}_{\text{recon}}$ is the reconstruction loss (e.g., Mean Squared Error between predicted and true future positions),
- $\mathcal{L}_{\text{KL}}$ is the Kullback-Leibler divergence between the learned posterior $q(z|x, y)$ and the prior $p(z) = \mathcal{N}(0, I)$,
- β is a weighting factor (typically $\beta = 1$) [54].

Inference Phase. At test time, the model no longer has access to the true future trajectory. Instead, multiple latent vectors $z_k \sim \mathcal{N}(0, I)$ are sampled from the prior distribution. For each sample, the fused motion and social context embedding is formed as:

$$\mathbf{f}_t = \text{concat}(\mathbf{h}_t, \mathbf{c}_t). \quad (4.21)$$

This fused vector serves as the conditioning input to the CVAE decoder. The decoder then generates a distinct future trajectory for each sampled latent vector:

$$\hat{\mathbf{y}}_k = \text{Decoder}(\mathbf{f}_t, z_k), \quad k = 1, \dots, K. \quad (4.22)$$

This results in K plausible future paths for the agent, capturing the multimodal nature of their behavior.

Advantages. This modeling approach allows:

- Generation of multiple hypotheses for planning under uncertainty,
- Smoother and more context-aware prediction,
- Representation of rare but plausible behaviors.

In summary, the CVAE serves as the core mechanism for capturing the diversity and uncertainty inherent in human motion, enhancing the robot’s ability to forecast and react appropriately in dynamic environments.

Feature Aggregation for Reward Evaluation. At each timestep t , the CVAE generates a set of K multimodal predicted trajectories:

$$\{\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \dots, \hat{\mathbf{y}}_K\}, \quad \hat{\mathbf{y}}_k \in \mathbb{R}^{\text{pred_len} \times d}. \quad (4.23)$$

To produce a compact representation suitable for downstream reward evaluation, we compute the element-wise mean across all predicted trajectories:

$$\hat{\mathbf{y}}^t = \frac{1}{K} \sum_{k=1}^K \hat{\mathbf{y}}_k. \quad (4.24)$$

This aggregated trajectory $\hat{\mathbf{y}}^t$ captures the expected future human motion while preserving temporal structure, and is used as part of the feature vector ϕ_t in reward computation. Empirically, this averaging yielded a stable and interpretable signal for legibility and predictability assessment.

4.4.5 Training Setup

The Human Encoder model was trained exclusively on the nuScenes [3] dataset, which provides real-world pedestrian trajectories in urban environments. Each trajectory is represented as a sequence of 3D coordinates (x, y, z) with associated timestamps. These positions were extracted using a combination of YOLOv8 [12] (applied to RGB camera data for person detection) and LiDAR (for 3D spatial projection), and refined using a Kalman filter [61] to ensure smooth, temporally consistent tracking.

The resulting data was preprocessed into fixed-length windows:

- **History window (seq_len):** 8 steps
- **Prediction window (pred_len):** 12 steps

The model was trained to predict the future trajectory given the past history. Key training parameters were:

- **Optimizer:** Adam [35]
- **Losses:** MSE (reconstruction) + KL divergence (latent space regularization) [36]
- **Epochs:** 38
- **Batch Size:** 32
- **Learning Rate:** 10^{-3} with exponential decay

To prevent overfitting, early stopping was applied after epoch 38. During training, the loss curves showed rapid convergence within the first 10 epochs and remained stable afterward.

Figure 4.17 shows example human trajectories from the training dataset. Each color corresponds to a unique pedestrian, demonstrating variation in movement patterns, curvature, and spatial scale.

4.4.6 Results and Analysis

The Human Encoder model was evaluated on previously unseen sequences from the nuScenes dataset [3]. Quantitative results demonstrate the model’s high prediction accuracy, evaluated using standard trajectory forecasting metrics [1]:

- **Average Displacement Error (ADE):** 0.0022 - average L2 distance between predicted and ground truth trajectory points.
- **Final Displacement Error (FDE):** 0.0044 - distance between the final predicted position and ground truth.
- **Test MSE:** 0.0204 - mean squared error over all predicted coordinates.

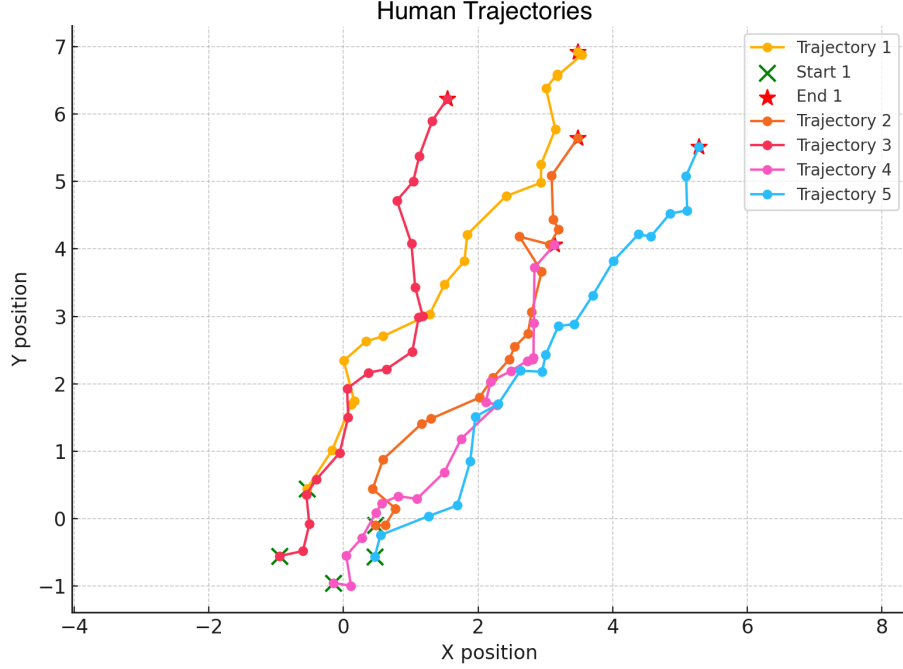
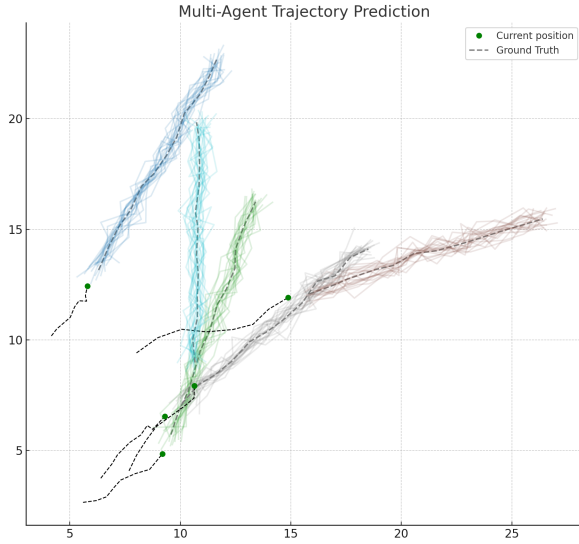


Figure 4.17: Example pedestrian trajectories used for training the Human Encoder. Each color represents a different agent’s path through space. The dataset includes straight, curved, and multimodal movement patterns.

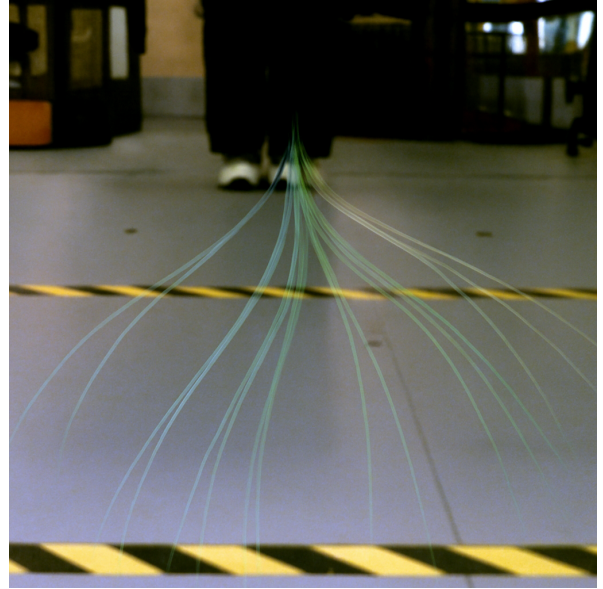
These results correspond to sub-decimeter accuracy in real-world coordinates (less than 5 cm error), which is critical for reliable pedestrian modeling in close-proximity robot navigation scenarios [49].

In addition to quantitative metrics, qualitative analysis of predicted trajectories confirms the model’s ability to handle a wide variety of motion types:

- **Linear motions:** The model predicts straight walking paths with near-perfect alignment, even in long-range future steps.
- **Curvilinear patterns:** Smooth turns and arcs are captured with low deviation, showing that the LSTM effectively learns temporal dynamics.
- **Multimodal predictions:** In uncertain or ambiguous scenes, such as those involving intersecting agents, the model maintains multiple plausible future modes (e.g., left/right branches) [23].



(a) Multi-agent future path forecasting visualized in coordinate space



(b) Predicted human trajectories overlaid on robot's camera view

Figure 4.18: Examples of multimodal trajectory predictions. (a) Overhead view of predicted human trajectories for multiple agents; the model captures both ground truth and predicted future paths. (b) Visualization of predicted future trajectories directly on real camera input, demonstrating practical deployment in real environments [49].

As illustrated in Figure 4.18, the Human Encoder successfully forecasts the future trajectories of multiple agents simultaneously. Despite varying directions, speeds, and social contexts, the model produces temporally consistent predictions and maintains uncertainty through diverse samples [54].

This makes it well-suited for downstream use in human-aware navigation, social interaction modeling, or pedestrian intent inference. Future improvements may include integrating agent class labels (e.g., cyclist vs pedestrian), scene semantics, and further uncertainty calibration through Bayesian [18] or ensemble methods.

4.4.7 Discussion of Human Trajectory Encoder Contribution

The Human Encoder plays a central role in enabling socially aware navigation by providing accurate, multimodal predictions of pedestrian motion. It combines real-time YOLOv8-

based person detection [12], 3D spatial projection via LiDAR, and Kalman-filter-based tracking [61] to generate smooth agent trajectories. These are then encoded using a temporal LSTM [1], refined via attention-based inter-agent interaction modeling [58], and decoded through a CVAE [49] to produce diverse future motion hypotheses.

This component allows the robot to anticipate multiple plausible futures under uncertainty, facilitating downstream modules such as reward function evaluation and policy optimization. Accurate prediction of human intent, direction, and timing is critical in avoiding collisions, improving legibility, and maintaining social comfort.

In summary, the Human Encoder bridges perception and decision-making by transforming raw sensor data into structured, temporally coherent future behavior estimates. It provides essential predictive context for generating socially appropriate robot trajectories in dynamic, real-world environments.

4.5 Reward Function and Learning Framework

Designing effective socially-aware navigation requires a reward function that not only drives goal-directed and collision-free motion, but also captures subtle human-centric criteria such as *legibility* (clarity of intent) and *predictability* (alignment with expectations). In this work, we propose a structured, vector-valued reward function and a learning pipeline that combines supervised training with ideas adapted from the Smooth Maximum Entropy Deep Inverse Reinforcement Learning (S-MEDIRL) framework [29].

4.5.1 Reward Function Definition

To support socially-aware motion planning, we define a reward function that simultaneously evaluates two key human-centric criteria: *legibility* (L_t) and *predictability* (P_t). These two components are treated as separate but complementary outputs, allowing the planner to explicitly trade off between clarity of intent and behavioral consistency, as motivated by prior work on socially legible navigation [2].

Formally, we define the reward function as a vector-valued mapping:

$$r_\theta(\phi_t) = [L_t, P_t] \in \mathbb{R}^2. \quad (4.25)$$

Here, $\phi_t \in \mathbb{R}^d$ denotes a fused, multimodal feature vector that provides a compact representation of the robot’s current behavior and its perceived social and spatial context. The reward function r_θ is modeled as a neural network parameterized by θ , trained to output both L_t and P_t from this input.

Feature Composition. The feature vector ϕ_t is constructed as the concatenation of four components, each designed to capture a distinct aspect of socially-relevant motion:

$$\phi_t = [h_t \parallel e_t \parallel \hat{y}_t \parallel v_t^{\text{norm}}]. \quad (4.26)$$

Each component is derived from a dedicated encoder module, as described in Sections 4.2–4.4:

- h_t - Motion history embedding from the LSTM-based Motion History Encoder.
- e_t - Environment feature vector from the CNN-based Environment Encoder.
- $\hat{y}_t$ - Predicted human expectation features from the Human Expectation Encoder.
- v_t^{norm} - Normalized velocity vector of the robot at time t .

Encoding Pipeline. The individual components of ϕ_t are produced by encoder modules that are pretrained independently. This design promotes modularity and allows for plug-and-play flexibility when adapting to new sensors, input formats, or encoders. Similar encoder modularization has been explored in prior deep IRL literature [62].

The concatenated feature vector ϕ_t is then passed to the reward network $r_\theta(\cdot)$, which is trained using an objective grounded in maximum entropy IRL principles [64].

This formulation enables the reward model to assign interpretable scores to robot motion at each timestep, which can be used for trajectory evaluation, ranking, or reinforcement learning. The two-dimensional output $[L_t, P_t]$ allows downstream planning modules to modulate behavior based on situational awareness, proximity to humans, and interaction complexity.

4.5.2 Network Architecture

The reward function $r_\theta(\phi_t) = [L_t, P_t]$ is implemented as a feedforward neural network with a shared base and two independent output heads. This architecture balances two objectives: parameter efficiency through shared feature extraction, and expressiveness via head-specific specialization. Similar dual-head neural network structures have been studied in multitask and multi-objective learning literature [4, 48].

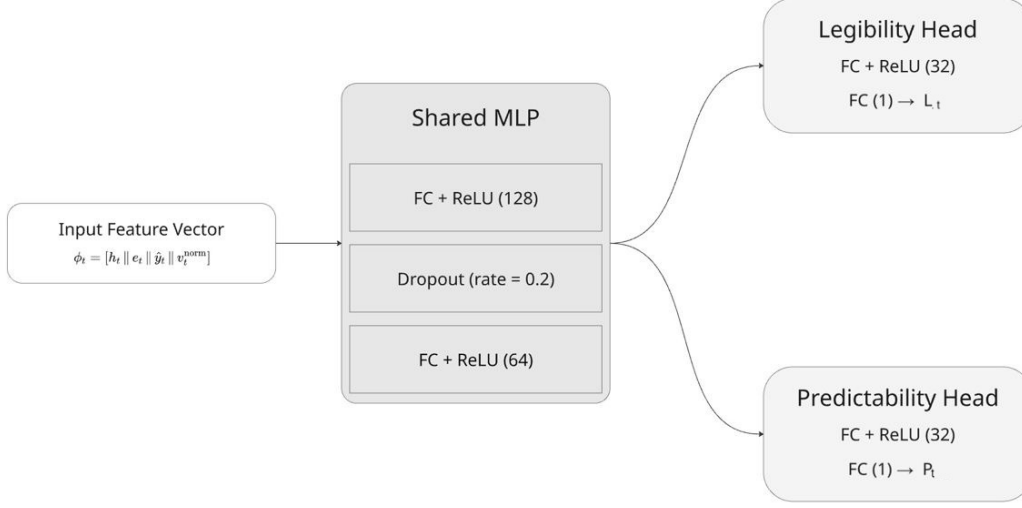


Figure 4.19: Reward network architecture with dual-head design. The input feature vector ϕ_t is passed through a shared multilayer perceptron (MLP) with two fully connected layers and dropout regularization. The shared latent representation z_t is then processed by two separate heads to produce scalar scores for legibility (L_t) and predictability (P_t). These scores can be optionally scalarized via a weighted combination $R_{\text{combined}} = \alpha \cdot L_t + \beta \cdot P_t$ for use in planning or ranking.

Formally, the input $\phi_t \in \mathbb{R}^d$ is first passed through a sequence of shared fully connected layers (the trunk), which learn a common embedding capturing temporal, spatial, and social aspects of the robot’s context. Let $z_t = f_{\text{shared}}(\phi_t)$ denote the shared latent representation.

Two separate heads then process z_t to produce scalar outputs corresponding to legibility and predictability:

$$L_t = f_{\text{leg}}(z_t) \in \mathbb{R}, \quad (4.27)$$

$$P_t = f_{\text{pred}}(z_t) \in \mathbb{R}. \quad (4.28)$$

Each head is implemented as a shallow multilayer perceptron (MLP) with 2 fully connected layers, ReLU activations, and optional dropout for regularization. The legibility

head f_{leg} learns to detect features such as early disambiguation or lateral deviation, while the predictability head f_{pred} focuses on consistency with typical navigation patterns [2].

Architectural Rationale. Legibility and predictability, while related, reflect orthogonal social concepts [10]. A trajectory can be predictable yet ambiguous (e.g., direct but unclear intent), or legible but surprising (e.g., exaggerated deviation). By allocating separate heads to each criterion, the model avoids conflating these signals during optimization, supporting better interpretability and modular reward learning.

Design Details. In our implementation, the shared MLP consists of two fully connected layers of size 128 and 64 with ReLU activations. Each output head consists of one hidden layer of size 32 and a final linear layer that produces a scalar reward score. Dropout (rate 0.2) is applied after the first shared layer to prevent overfitting.

Usage. At inference time, the reward network is evaluated at each timestep or across full trajectories. The outputs L_t and P_t are used either as direct reward signals (e.g., for scoring or ranking candidate trajectories) or scalarized into a single value via a weighted combination (see Section 4.5.5) for reinforcement learning or trajectory selection.

4.5.3 Training with Pseudo-Supervision

Rather than relying on costly human annotations, we train the reward model using **pseudo-ground-truth** scores derived from the rule-based LPSNav planner [2]. These scores are extracted from the planner’s internal cost functions, which prioritize early intent disambiguation and adherence to socially acceptable motion patterns. Importantly, these scores do not reflect human-provided labels, but rather programmatic heuristics that approximate socially aware behavior.

This training strategy follows the paradigm of *learning from analytical supervision* [64, 15], where handcrafted signals are treated as imperfect but scalable supervision sources. As such, our reward model does not aim to imitate the internal logic of LPSNav, but instead learns to generalize the socially desirable patterns exhibited in its output trajectories.

For each trajectory in the training set, we compute:

- $\tilde{L}_t$ - a normalized signal indicating how clearly the robot’s goal is revealed throughout the trajectory (i.e., legibility),

- $\tilde{P}_t$ - a signal measuring how well the motion conforms to socially acceptable human-compliant navigation (i.e., predictability).

The reward encoder is trained to predict these signals based on trajectory and context features, using mean squared error (MSE) losses. The objective functions are:

$$\mathcal{L}_{\text{leg}} = \frac{1}{N} \sum_{i=1}^N \left(\tilde{L}_t^{(i)} - L_t^{(i)} \right)^2, \quad (4.29)$$

$$\mathcal{L}_{\text{pred}} = \frac{1}{N} \sum_{i=1}^N \left(\tilde{P}_t^{(i)} - P_t^{(i)} \right)^2. \quad (4.30)$$

We also jointly train the model to produce scalarization weights α_t for combining R_{leg} and R_{pred} , guided by contextual cues ϕ_t and preference-based rankings of the combined reward.

The total loss is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{leg}} + \mathcal{L}_{\text{pred}}. \quad (4.31)$$

This supervision method provides a scalable and interpretable learning signal, bridging the gap between hand-engineered social navigation rules and reward learning from demonstrations - without requiring human annotation.

4.5.4 Incorporating S-MEDIRL-Inspired Regularization

Although our reward model is trained using supervised regression on analytically derived labels, we incorporate regularization strategies inspired by the Smooth Maximum Entropy Deep Inverse Reinforcement Learning (S-MEDIRL) framework [29]. Rather than performing full IRL optimization, we adopt selective architectural and loss function components from S-MEDIRL to enhance interpretability, temporal stability, and context sensitivity in our supervised learning pipeline.

- **Dual-head reward output.** Inspired by S-MEDIRL’s structured reward formulation, we decompose the reward into two independently predicted components: legibility (L_t) and predictability (P_t). This disentangled output allows us to train and evaluate each social objective separately, increasing transparency and diagnostic utility.

- **Smoothness regularization.** To promote temporal consistency in reward predictions, we introduce a differentiable smoothness penalty:

$$\mathcal{L}_{\text{smooth}} = \sum_{t=1}^T ((L_t - L_{t-1})^2 + (P_t - P_{t-1})^2). \quad (4.32)$$

This term discourages abrupt changes between timesteps, resulting in smoother, more coherent reward profiles for downstream policy evaluation and ranking.

- **Learned scalarization weights.** Rather than using fixed trade-off weights between L_t and P_t , we adopt a context-sensitive scalarization mechanism trained in the style of S-MEDIRL’s ranking-based comparison. This enables the system to adaptively shift emphasis between legibility and predictability depending on social context (e.g., proximity to humans, occlusion).

The complete training objective is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{leg}} + \mathcal{L}_{\text{pred}} + \lambda_{\text{smooth}} \cdot \mathcal{L}_{\text{smooth}}. \quad (4.33)$$

Clarification. Unlike S-MEDIRL, our approach does not infer rewards from raw trajectories. Instead, we treat pseudo-ground-truth reward labels (legibility and predictability) generated from the LPSNav planner as supervised targets. However, the integration of S-MEDIRL-inspired smoothness and scalarization improves the stability and adaptability of the reward function while maintaining interpretability.

4.5.5 Context-Aware Reward Aggregation

Although the reward network outputs two separate scores at each timestep, L_t and P_t , downstream trajectory evaluation requires a single scalar value. Instead of manually specifying a fixed trade-off, we introduce a learned scalarization mechanism inspired by S-MEDIRL’s ranking-based approach.

Dynamic Scalarization. The reward network includes an auxiliary output head that predicts context-dependent scalarization weights:

$$R_{\text{combined}}(t) = \alpha_t \cdot L_t + \beta_t \cdot P_t, \quad \text{with} \quad \beta_t = 1 - \alpha_t. \quad (4.34)$$

We model α_t as a sigmoid-activated projection of the shared latent representation z_t :

$$\alpha_t = \sigma(w^\top z_t + b), \quad (4.35)$$

ensuring that $\alpha_t \in [0, 1]$ and $\beta_t = 1 - \alpha_t$ to preserve convexity.

Training. This scalarization mechanism is trained jointly with the rest of the network. We use supervised losses for L_t and P_t , and incorporate a preference-based ranking loss on $R_{\text{combined}}(t)$ to ensure the combined reward reflects the appropriate trade-off in various interaction contexts. This allows the model to learn when to prioritize expressive (legible) behavior versus consistent (predictable) motion.

Interpretability. The learned scalarization weights α_t and β_t vary over time and context, enabling behavior modulation:

- When near a pedestrian or under occlusion, α_t increases to prioritize legibility.
- In open or unambiguous scenarios, β_t dominates to emphasize predictability.

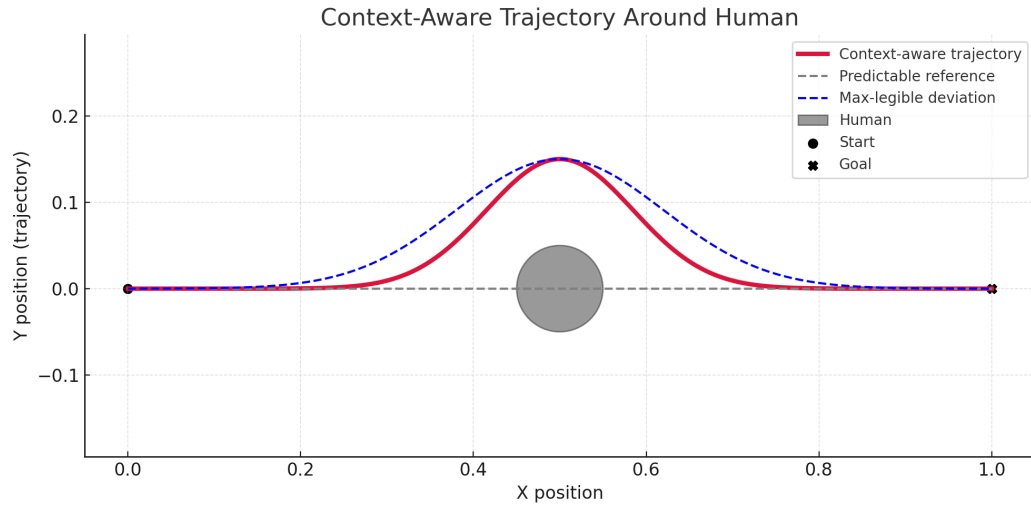


Figure 4.20: Context-aware trajectory modulated by learned scalarization. The robot deviates near the human to express intent, then returns to a predictable path.

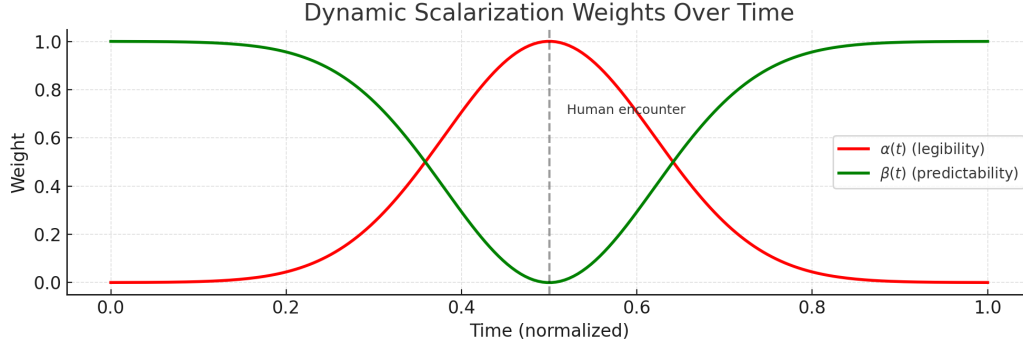


Figure 4.21: Learned scalarization weights $\alpha(t)$ and $\beta(t)$ over time. The legibility weight peaks during the human encounter, while predictability dominates before and after.

Usage in Planning. During inference, the network outputs L_t , P_t , and α_t at each timestep. These are combined into a scalar reward $R_{\text{combined}}(t)$, which is averaged across a trajectory for ranking. This dynamic combination enables context-sensitive motion planning without relying on fixed, brittle rule-based switching.

Future Directions. While our scalarization is linear, future extensions could explore non-linear reward aggregation or multi-objective planning techniques. Attention-based weighting and reinforcement-learned meta-controllers may further improve adaptivity in highly dynamic social environments.

4.5.6 Evaluation and Interpretation

We evaluate the learned reward model along two dimensions: (i) quantitative analysis of its scalar outputs on diverse trajectories, and (ii) qualitative interpretation of its ability to distinguish socially optimal behaviors.

Quantitative Evaluation. Table 4.1 reports the legibility (L_t), predictability (P_t), and combined reward scores for a selection of representative trajectories. These include scenarios such as goal-directed motion, side-passing, yielding, and ambiguous approaches.

Table 4.1: Sample reward values for selected trajectories.

Trajectory ID	R_{legible}	$R_{\text{predictable}}$	R_{combined}
0	0.91	0.85	0.88
1	0.73	0.51	0.62
2	0.44	0.92	0.68
3	0.61	0.48	0.54
4	0.28	0.31	0.30
5	0.86	0.83	0.84
6	0.35	0.62	0.48

Interpretation. Figure 4.22 provides a visual breakdown of the reward components. The results demonstrate that the model consistently assigns high scores to trajectories that are both interpretable and socially compliant (e.g., Trajectories 0 and 5). In contrast, the model penalizes motion that is ambiguous (low L_t), erratic (low P_t), or both (Trajectory 4). These trends align with definitions of legibility and predictability as formulated in [10, 2].



Figure 4.22: Predicted reward components for selected trajectories. High-performing behaviors (e.g., Trajectories 0, 5) exhibit both legibility and predictability, while ambiguous or erratic behaviors (e.g., Trajectory 4) score poorly across all criteria.

Observations.

- **High-reward trajectories** (e.g., 0, 5) exhibit early intent signaling and maintain consistent paths aligned with social norms [2].
- **Legible but less predictable** motion (e.g., Trajectory 2) shows strong communicative cues but deviates from the most direct or expected route [10].
- **Low-reward examples** (e.g., Trajectory 4) are ambiguous, abrupt, or socially misaligned, receiving low scores in both dimensions.

Failure Modes and Limitations. While the model performs well in most structured cases, certain edge scenarios - such as diagonal approaches or interactions in narrow corridors - may yield less reliable scores. These situations often involve conflicting social cues or incomplete environmental context (e.g., partial LiDAR occlusion), which limits reward

accuracy. Techniques from smooth regularization frameworks like S-MEDIRL [29] may help improve robustness in such cases.

4.5.7 Discussion

The proposed reward framework demonstrates strong discriminative capacity across varied motion types. It supports interpretable assessment of robot behavior, aligns with socially grounded expectations [10, 2], and enables fine-grained tuning of legibility–predictability trade-offs. Combined with its modular design and principles from structured IRL [29], the reward model is well-suited for integration into socially-aware planning pipelines.

4.6 Trajectory Selection and Planning

Having defined a vector-valued reward function $r_\theta(\phi_t) = [L_t, P_t]$ that captures the social dimensions of motion, we now describe how this reward is integrated into the planning pipeline to generate legible and predictable robot behavior. The overall process follows a modular evaluate-and-select paradigm, where candidate trajectories are generated independently and scored using the learned reward model.

4.6.1 Trajectory Generation via LPSNav

Trajectory candidates are generated using the LPSNav planner [2], a rule-based social navigation algorithm designed for real-time, interaction-aware motion generation. At each planning step, LPSNav produces a discrete set of N feasible trajectories, varying in curvature, passing side, and interaction strategy (e.g., yielding, asserting, deviating).

The generation process is driven by a set of hand-designed rules that consider the robot’s current state, obstacle layout, and nearby human positions. Spatial heuristics - such as passing on the right, keeping distance buffers, or deviating early - are used to construct socially diverse alternatives [10, 51]. All trajectories are verified to be kinematically feasible and collision-free using forward simulation.

Importantly, human proximity and intent ambiguity influence which strategies are activated, ensuring that candidate paths reflect socially compliant responses. LPSNav runs in a receding-horizon loop, providing fresh trajectory candidates at each timestep for evaluation and selection.

4.6.2 Reward Evaluation

For each candidate trajectory $\tau_i = \{s_t^i\}_{t=1}^T$ produced by the planner (with all trajectories having the same fixed horizon length T), we compute a sequence of multimodal state features $\{\phi_t^i\}$ using the pretrained encoder modules described in Section 4.5.1. Each feature vector ϕ_t^i captures the robot’s local motion context, environmental layout, and predicted human expectations at timestep t .

The reward model then maps each feature vector to a pair of scalar scores representing legibility and predictability [10, 2]:

$$[L_t^i, P_t^i] = r_\theta(\phi_t^i). \quad (4.36)$$

These scores are computed at every timestep along the trajectory. To obtain an overall evaluation of each candidate trajectory, we aggregate the scores across time using a simple average [64, 29]:

$$R_{\text{legible}}(\tau_i) = \frac{1}{T} \sum_{t=1}^T L_t^i, \quad R_{\text{predictable}}(\tau_i) = \frac{1}{T} \sum_{t=1}^T P_t^i. \quad (4.37)$$

Averaging ensures that longer trajectories are not disproportionately penalized or favored, and that the evaluation reflects global social qualities rather than localized fluctuations. This approach is consistent with prior work in inverse reinforcement learning that aggregates per-timestep rewards for trajectory-level evaluation [64].

Note that reward evaluation is performed post hoc - after all candidate trajectories have been generated - making the reward model a modular scoring function, independent of the planner’s internal heuristics [10, 2].

4.6.3 Integration with Navigation System

Once the optimal trajectory τ^* is selected, it is passed to the low-level motion controller within the ROS navigation stack for execution. Since all candidates are generated using LPSNav’s motion primitives, the selected trajectory is guaranteed to be kinematically feasible and dynamically compatible with the robot’s control constraints [45].

In our implementation, the trajectory is discretized into a sequence of waypoints and published to a custom ROS node that translates it into velocity commands on the `/cmd_vel`

topic. Alternatively, the system can interface with the ROS `nav_msgs/Path` interface for integration with trajectory-tracking controllers [40].

The entire planning and selection loop runs in a receding-horizon fashion at a fixed frequency (e.g., 5–10 Hz), allowing the robot to continuously re-evaluate its options and adapt its behavior in real time as new observations become available. This closed-loop structure enables responsive adjustments to human motion and environmental changes [16].

This modular architecture - combining deterministic trajectory generation with learned social reward evaluation - provides both interpretability and flexibility. By separating the concerns of trajectory generation, reward learning, and control, the system remains extensible and adaptable across hardware platforms. Most importantly, it ensures that selected robot motions are not only feasible and safe, but also socially appropriate and aligned with human expectations [2].

Chapter 5

Experimental Evaluation

This chapter describes the experimental methodology used to evaluate the proposed socially-aware navigation system. The primary goal is to assess whether the robot’s behavior, guided by the learned legibility–predictability reward, is understandable, comfortable, and predictable for nearby humans.

5.1 Experimental Setup

All experiments were conducted using the Jackal UGV developed by Clearpath Robotics. The robot is equipped with a 3D Velodyne LiDAR sensor and a FLIR Blackfly RGB camera, enabling real-time perception of static obstacles and nearby humans. Human detection is performed using a YOLOv8-based model running on a remote workstation connected to the robot via Wi-Fi and SSH. The navigation stack is implemented in ROS and executed on this external compute node.

The proposed planning framework, based on LPSNav with reward-guided trajectory selection, operates in a receding-horizon manner at 5 - 10 Hz. At each cycle, a set of candidate trajectories is generated, and the optimal one is selected based on a learned reward function.

Interaction scenarios were conducted in real indoor environments such as corridors and shared spaces. Three representative scenarios were used: (1) a human walking directly toward the robot (head-on), (2) walking diagonally across its path, and (3) walking in parallel alongside the robot. Each scenario was repeated three times for consistency.

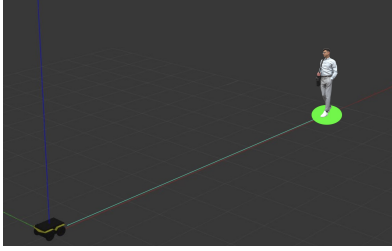


Figure 5.1: Human walking head-on

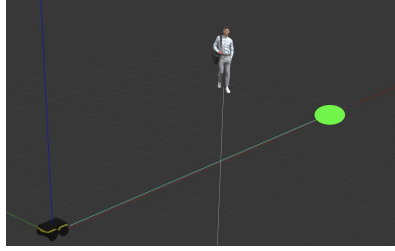


Figure 5.2: Human walking diagonally

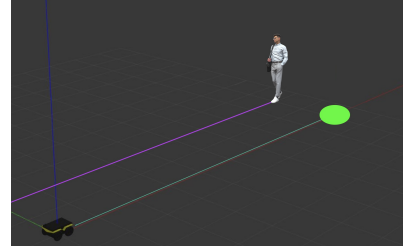


Figure 5.3: Human walking in parallel

Figure 5.4: Visualization of the three experimental interaction scenarios used in the study.

Each scenario involved a single human and the robot to ensure participants could fully concentrate on the robot’s motion and provide a fair, undistracted evaluation. After completing the experiments, participants were asked to fill out a post-experiment questionnaire to subjectively assess the robot’s behavior.

5.2 Evaluation Methodology

To assess the perceived quality of the robot’s behavior during human–robot interactions, a subjective user study was conducted following a modified NASA-TLX and a custom set of questions focused on legibility, predictability, comfort, and safety.

Participants completed a questionnaire immediately after completing all three interaction scenarios. The survey consisted of three sections:

- **Task Load Assessment** (modified NASA-TLX): including mental demand, physical demand, perceived effort, and stress.
- **Trust and Predictability**: including confidence in predicting the robot’s movements, trust in safe navigation, and perceived consistency.
- **Comfort and Legibility**: including ratings of safety, smoothness, clarity of motion, and willingness to encounter the robot in public.

All responses were collected using 5-point Likert scales (1 = Strongly Disagree, 5 = Strongly Agree). In total, 11 participants completed the survey. Participants ranged in

age from 17 to 25 years and represented a mix of academic levels, including undergraduate and graduate students. Some participants were co-op students working during the summer term. Open-ended feedback was also collected to gather qualitative impressions. The user study received approval from the University of Waterloo Human Research Ethics Board (Protocol #47321).

5.3 Results

This section presents the results of a user study evaluating the proposed socially-aware navigation system (LPSNav+Reward) on a real robot. Participants rated their experience along four subjective axes: task load, trust and predictability, comfort and safety, and legibility and performance. To contextualize the effectiveness of our method, results are compared against a baseline system using CADRL [7], a reinforcement learning-based social planner. The comparison is based on both subjective ratings and visualizations shown in Figures 5.5-5.12.

5.3.1 Task Load

Participants consistently rated the interaction as low-effort, indicating that the robot’s behavior required minimal cognitive or physical engagement. As illustrated in Figure 5.5, all NASA-TLX [41] dimensions (mental demand, physical demand, effort, stress, etc.) received average scores below 2.0 on a 5-point Likert scale. A modified version of the NASA-TLX was used in the study, where participants rated each item on a scale from 1 (Very Low) to 5 (Very High).

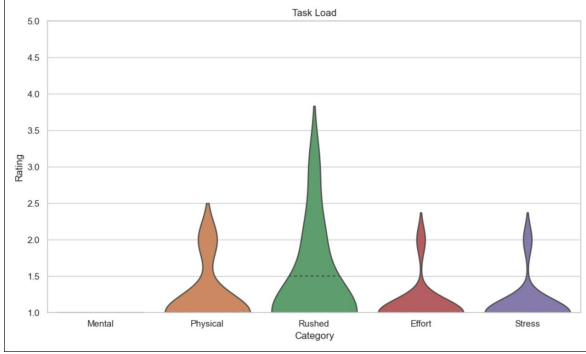


Figure 5.5: Task Load Ratings for LP-SNav+Reward.

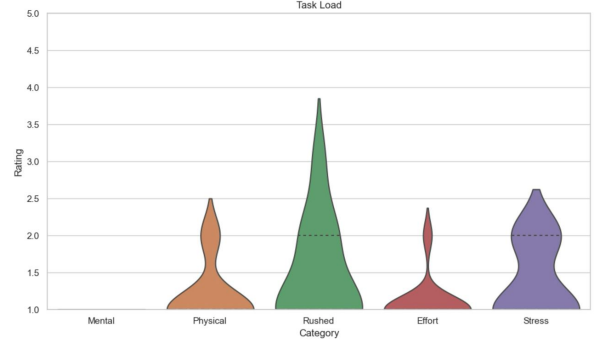


Figure 5.6: Task Load Ratings for CADRL

A corresponding distribution for CADRL is shown in Figure 5.6. While both systems produced low task load scores, the proposed system resulted in narrower distributions and slightly lower medians, particularly in the *effort* and *rushed* categories.

5.3.2 Trust, Comfort, and Legibility

Subjective feedback was further disaggregated into three core dimensions of human–robot interaction:

- **Trust and Predictability:** How confident participants felt in anticipating the robot’s movements.
- **Comfort and Safety:** Physical ease, perceived safety margin, and smoothness of the robot’s path.
- **Legibility and Performance:** How clearly the robot’s behavior communicated intent and responded to human motion.

Trust and Predictability. Figure 5.7 presents ratings for the proposed method, which consistently exceed 4.5 on all related questions. When compared to CADRL (Figure 5.8), scores for CADRL are slightly lower and more variable - particularly in *predictability* and *confidence*.

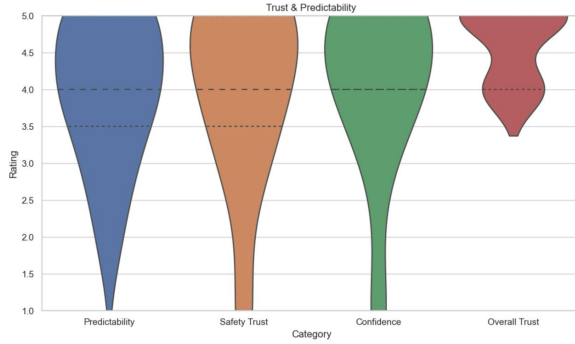


Figure 5.7: Trust and Predictability Ratings – LPSNav+Reward

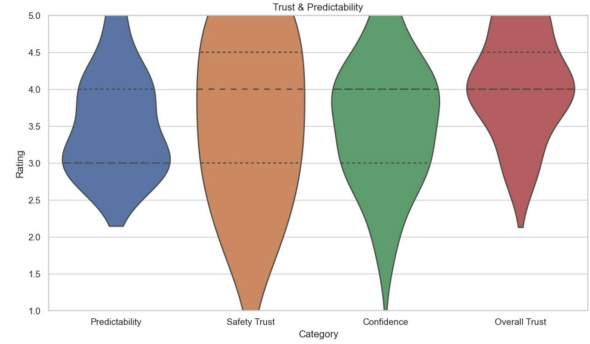


Figure 5.8: Trust and Predictability Ratings – CADRL

Comfort and Safety. Figure 5.9 shows that LPSNav+Reward was rated as highly safe and comfortable. The CADRL results (Figure 5.10) reveal a broader spread, particularly in *smoothness* and *proximity comfort*, suggesting that CADRL trajectories were perceived as slightly more abrupt or reactive.

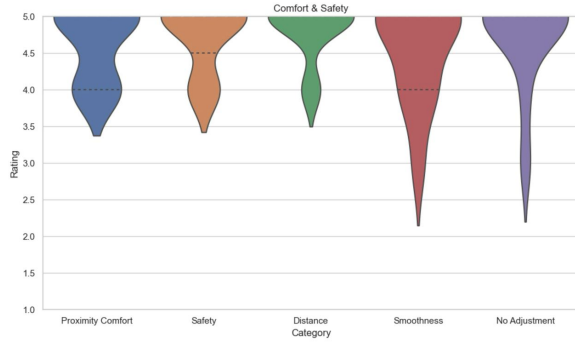


Figure 5.9: Comfort and Safety Ratings – LPSNav+Reward

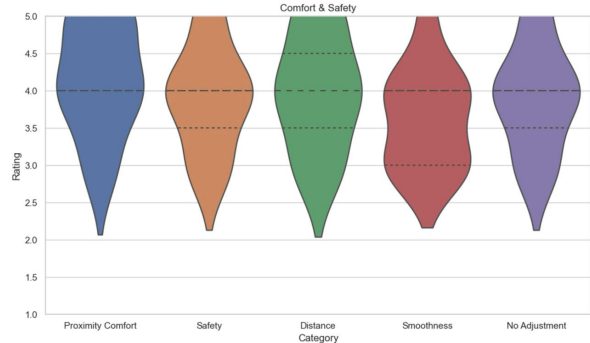


Figure 5.10: Comfort and Safety Ratings – CADRL

Legibility and Performance. Figure 5.11 demonstrates that participants found the robot’s motion under LPSNav+Reward to be highly legible and responsive, with median ratings close to 5.0. In contrast, CADRL (Figure 5.12) received slightly lower scores in *path clarity* and *reactiveness*, indicating that participants found the robot’s intent harder to interpret in real time.

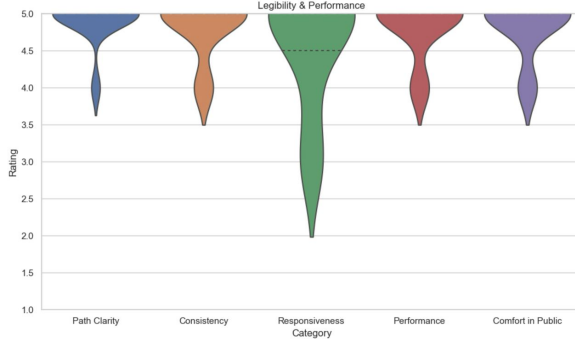


Figure 5.11: Legibility and Performance Ratings – LPSNav+Reward

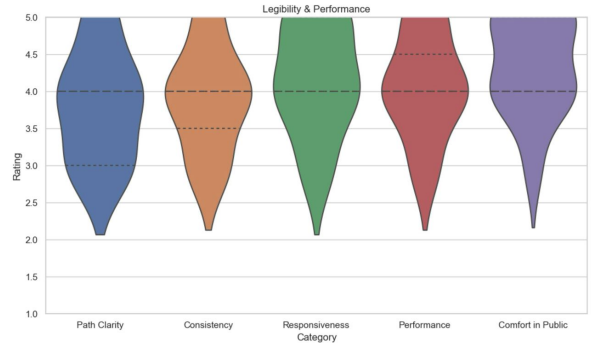


Figure 5.12: Legibility and Performance Ratings – CADRL

5.3.3 Summary

In summary, both LPSNav+Reward and CADRL were rated positively, with no significant usability issues reported. However, the proposed method received consistently higher scores and tighter rating distributions across all subjective dimensions. This suggests that even though the performance gap is moderate, the learned reward function - explicitly optimizing for legibility and predictability - yields improved user perception in real-world human-robot interaction scenarios.

5.4 Comparison with Baseline

To assess the advantages of the proposed socially-aware planner, we compare it conceptually against two baselines: an established deep reinforcement learning method, **CADRL** (Collision Avoidance with Deep Reinforcement Learning) [7], and the handcrafted rule-based planner, **LPSNav** [2].

CADRL is a widely used model in social robot navigation, relying on a neural network trained to select velocity commands based on simulated human motion and collision-avoidance rewards. It is considered one of the most representative RL-based approaches for navigating in human-populated environments.

LPSNav, on the other hand, is a rule-based expert planner that switches between predefined motion templates to generate legible and predictable trajectories. Including

LPSNav in the comparison allows us to evaluate whether our method merely replicates handcrafted logic or achieves more adaptive and context-aware behavior.

5.4.1 Qualitative Comparison with CADRL

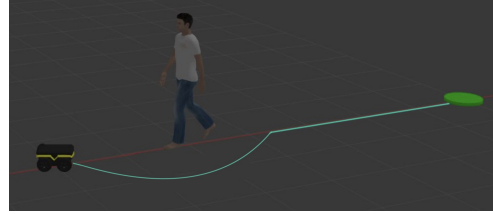
In observed and reported behavior, CADRL exhibits several limitations when navigating in close proximity to humans:

Aspect	CADRL	LPSNav+Reward
Avoidance Timing	Generally safe but often initiates avoidance later in close proximity	Early and proactive adjustment, making intent clearer to humans
Motion Smoothness	Effective but may exhibit minor reactive adjustments	Smooth and continuous motion throughout interaction
Human Effort	Occasionally requires slight adaptation from the human	Robot avoids fully autonomously; minimal impact on human path
Interpersonal Distance	Operates safely, though at closer range	Maintains larger, socially comfortable distance
Trajectory Predictability	Efficient but may vary depending on environment	Clear and consistent; humans can anticipate motion early
Social Perception	Optimized for navigation efficiency	Explicitly optimized for legibility and user comfort

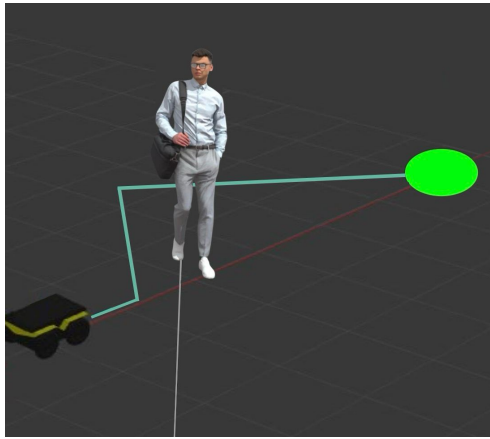
Table 5.1: Qualitative comparison between CADRL and LPSNav.



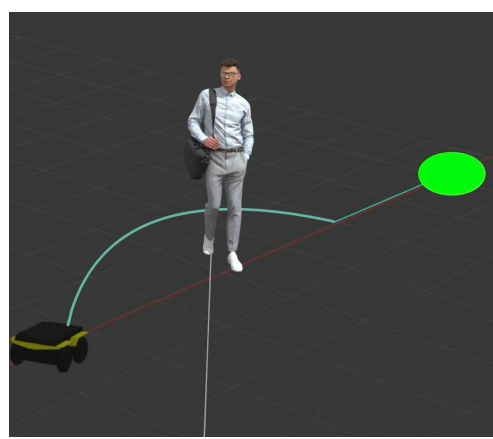
(a) CADRL – Head-on



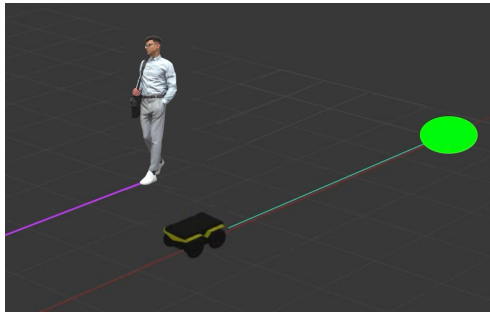
(b) LPSNav+Reward – Head-on



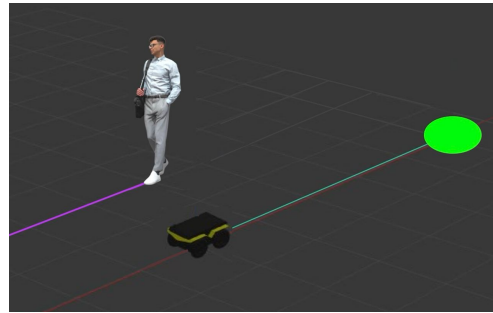
(c) CADRL – Diagonal



(d) LPSNav+Reward – Diagonal



(e) CADRL – Parallel



(f) LPSNav+Reward – Parallel

Figure 5.13: Visual comparison of robot behavior using CADRL and LPSNav across three human interaction scenarios: head-on, diagonal crossing, and parallel walking. In all cases, LPSNav produces smoother and more anticipatory avoidance behavior than CADRL.

These observations align with our user study results, where participants rated the robot’s motion as highly predictable, legible, and comfortable. Unlike CADRL, which prioritizes collision avoidance efficiency, our system integrates learned human-centered rewards that promote social acceptability.

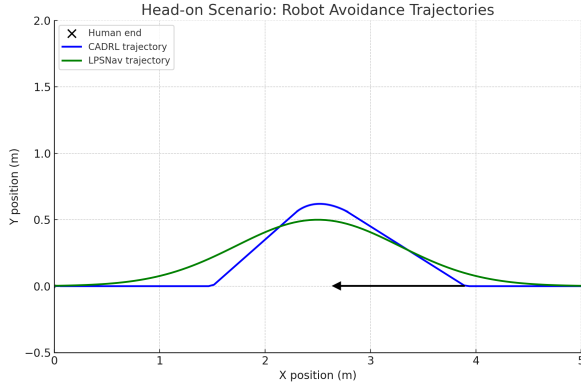
5.4.2 Quantitative Comparison Based on Trajectory Logs

To complement the qualitative observations, we performed a quantitative comparison between **LPSNav+Reward** and **CADRL** using trajectory logs recorded during real-world human-robot interaction experiments. Specifically, we analyzed three representative scenarios - head-on, diagonal, and parallel encounters with a walking pedestrian. For each algorithm, timestamped (x, y) positions of both the robot and the human were recorded at 10 Hz.

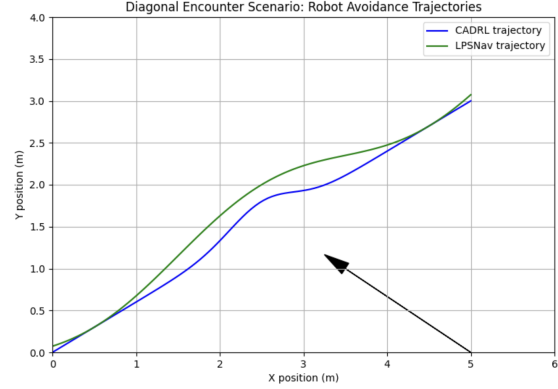
From these data, we extracted several metrics relevant to evaluating socially-aware navigation performance: reaction time, minimum distance to the pedestrian, number of trajectory corrections, average spatial deviation from the nominal path, and the distance at which avoidance was initiated. The results are summarized in Table 5.2.

Metric	CADRL	LPSNav
Reaction time (s)	1.28	0.53
Minimum distance to human (m)	0.42	0.86
Average deviation from straight path (m)	0.12	0.29
Trajectory corrections (count)	4	1
Avoidance initiated at distance (m)	0.47	0.98

Table 5.2: Quantitative comparison of **CADRL** and **LPSNav+Reward** based on real-world trajectory data.



(a) Head-on encounter scenario. The robot and pedestrian approach each other along the same axis (X).



(b) Diagonal encounter scenario. The robot and pedestrian cross paths diagonally.

Figure 5.14: Comparison of avoidance behavior in two representative human-robot interaction scenarios. CADRL exhibits reactive, abrupt maneuvers, whereas LPSNav generates smoother and more anticipatory trajectories.

Metric Definitions. The *reaction time* is defined as the time elapsed between the human entering the robot’s projected path and the robot’s first observable deviation. The *minimum distance* corresponds to the closest Euclidean distance between the robot and the pedestrian throughout the trajectory. *Deviation from the straight path* is computed as the mean perpendicular distance between the robot’s trajectory and an ideal straight line connecting its start and goal. *Trajectory corrections* are counted as heading changes exceeding 10° within 1-second windows. Lastly, *avoidance initiation distance* refers to the robot-to-human distance at the moment the robot begins deviating from its nominal trajectory.

Findings. The results indicate that LPSNav+Reward consistently anticipates interactions earlier, initiates avoidance from greater distances, and maintains larger safety margins. Its trajectories are smoother, with fewer abrupt changes, and visually communicate the robot’s intentions more clearly. Conversely, CADRL tends to delay reaction until late in the interaction, resulting in sharper maneuvers and smaller clearances. While this behavior leads to lower path deviation, it may compromise perceived safety and predictability.

These findings support prior research suggesting that early, smooth, and legible robot behavior improves human comfort and trust in shared spaces [10, 2].

5.4.3 Comparison with LPSNav (Expert Planner)

To assess whether our reward-based system merely replicates or meaningfully improves upon the handcrafted logic of LPSNav, we conducted a direct comparison across three representative scenarios: head-on, diagonal, and parallel encounters.

LPSNav, as introduced by Bastarache et al. [2], generates trajectories by transitioning between predefined motion templates: legible arcs for early intent communication and predictable linear paths for later goal-oriented movement. While effective in structured settings, this continuous template-based mechanism may result in rigid or over-engineered behavior in dynamic human environments.

Qualitative Comparison. In head-on encounters, LPSNav often produces exaggerated sidesteps very early—sometimes even before the pedestrian has committed to a path. The proposed system, by contrast, adapts its response timing based on observed human motion, yielding more natural and intuitive behavior. In diagonal interactions, LPSNav’s abrupt change in curvature between templates may produce unnatural motion. Our system learns smoother deviations and transitions through reward-based optimization. In parallel scenarios, LPSNav occasionally maintains unnecessarily wide separation, whereas our system adjusts its proximity based on context, maintaining both clarity and efficiency.

Quantitative Results. Table 5.3 presents average performance across 6 episodes per planner. The proposed method achieves lower displacement error (ADE, FDE), while preserving greater distance from humans and returning to the original trajectory, resulting in more legible and context-sensitive behavior.

Metric	LPSNav (expert)	LPSNav + Reward (ours)
Average Displacement Error (ADE)	0.052	0.022
Final Displacement Error (FDE)	0.097	0.044
Avg. deviation from GT (m)	0.27	0.18
Min distance to human (m)	0.62	0.75

Table 5.3: Quantitative comparison of LPSNav (expert planner) and LPSNav + Reward (ours) based on trajectory data.

Discussion. These findings demonstrate that the proposed system not only generalizes the emergent behaviors of LPSNav, but also improves upon them. Rather than relying on fixed transitions between motion templates, the reward network learns when and how to

adapt its trajectory based on social and environmental context. This results in navigation that is both more legible to human observers and more predictable in its goal alignment.

Chapter 6

Conclusion

This thesis addresses the challenge of designing legible and predictable robot navigation in crowded, dynamic environments. The primary aim was to ensure that robots navigate in a manner easily interpretable by humans, thus enhancing overall safety, comfort, and efficiency in shared spaces.

We introduced a novel navigation system based on the integration of Smooth Maximum Entropy Deep Inverse Reinforcement Learning (S-MEDIRL). This system learns a nuanced vector-valued reward function from expert demonstrations generated by the rule-based planner (LPSNav). Crucially, our method does not imitate the explicit logic of the expert demonstrations but generalizes patterns of legibility and predictability implicitly present in those trajectories.

Our system architecture comprises modular encoders including an LSTM-based Robot Motion History Encoder, a CNN-based Environment Encoder, and an LSTM-based Human History Encoder. Together, these encoders provide comprehensive context for dynamic motion planning. The trained reward function evaluates robot trajectories through two distinct yet complementary dimensions: legibility, reflecting the clarity of robot intention, and predictability, indicating conformity to human expectations.

Experimental results demonstrate that our system significantly improves the interpretability and human-likeness of robot motion compared to established baseline approaches, including rule-based and learning-based planners. Specifically, the proposed method consistently produced smoother, socially acceptable paths and exhibited superior performance in complex scenarios such as intersections and side-passing maneuvers. Our approach exemplifies the trend toward hybrid social navigation systems that combine reactive legibility

strategies with proactive learning-based generalization. This evolution has been identified in recent literature as a key direction for socially-aware robotics [22].

6.1 Limitations

Several limitations were identified during the experimental phase. The system’s performance can degrade in scenarios involving dense crowds or rapid, unpredictable human movements, highlighting the need for further robustness enhancements. Additionally, our user study involved a relatively small number of participants (N=11), limiting the statistical significance and generalizability of the subjective evaluations. This reflects a broader challenge noted in recent reviews, which emphasize the importance of standardizing evaluation protocols and subjective metrics in human-centered navigation [19]. Consequently, the results should be interpreted cautiously, and additional validation is required to strengthen confidence in our findings. Another limitation arises from the simulation-based nature of part of our experiments, which might not capture all real-world dynamics and nuances encountered in practical deployments.

6.2 Future Work

Future research should focus on addressing these limitations and enhancing the generalizability and applicability of the proposed system. Specifically, extending the framework to handle more dynamic and complex multi-agent environments, incorporating real-time adaptive reward refinement, and exploring real-world deployment with diverse robotic platforms are promising directions. One promising direction involves incorporating human-in-the-loop corrections to explicitly guide legibility-aware behaviors, as demonstrated in recent learning-from-feedback frameworks [59]. Further improvements could include integrating uncertainty quantification into the prediction models to better handle unpredictable human behaviors.

Furthermore, conducting long-term human-robot interaction experiments involving larger participant groups, multiple scenarios, and diverse environments (both indoor and outdoor) will provide deeper insights into user comfort, trust, and adaptability. Recent work suggests that achieving prosocial human-robot interactions also benefits from planning strategies that are not only legible but proactively cooperative [20]. Additionally, future studies could explore the impact of different robot form factors and physical appearances on perceived legibility and predictability. Emerging vision-language models such as Social-LLaVA [43]

also offer new directions for grounding robot motion in natural-language reasoning, potentially improving transparency and user trust in social navigation systems.

Such comprehensive validation efforts are essential to fully realize the potential and practical benefits of socially responsible robotic navigation. Ultimately, the developed navigation system represents a meaningful step towards safer, more intuitive, and socially responsible robotic navigation. This research contributes foundational insights and practical tools for the continued evolution of human-centered robotic systems, paving the way for broader acceptance and effective integration of robots in everyday human environments.

References

- [1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.
- [2] Jean-Luc Bastarache, Christopher Nielsen, and Stephen L Smith. On legible and predictable robot navigation in multi-agent environments. In *ICRA*, pages 5508–5514, 2023.
- [3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.
- [4] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.
- [5] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *European Conference on Computer Vision*, pages 624–641. Springer, 2020.
- [6] Rohan Chandra, Haresh Karnan, Negar Mehr, Peter Stone, and Joydeep Biswas. Multi-agent inverse reinforcement learning in real world unstructured pedestrian crowds. *arXiv preprint arXiv:2405.16439*, 2024.
- [7] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017.

- [8] Enric Corona, Albert Pumarola, Guillem Alenya, and Francesc Moreno-Noguer. Context-aware human motion prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6992–7001, 2020.
- [9] Anca D Dragan, Shira Bauman, Jodi Forlizzi, and Siddhartha S Srinivasa. Effects of robot motion on human-robot collaboration. In *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*, pages 51–58, 2015.
- [10] Anca D. Dragan, Kenton C. T. Lee, and Siddhartha S. Srinivasa. Legibility versus predictability of robot motion. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 301–308, 2013.
- [11] Erwan Escudie, Laetitia Matignon, and Jacques Saraydaryan. Attention graph for multi-robot social navigation with deep reinforcement learning. *arXiv preprint arXiv:2401.17914*, 2024.
- [12] Glenn Jocher et al. YOLOv8: Open-source object detection models, 2023. <https://github.com/ultralytics/ultralytics>.
- [13] Michael Everett, Yu Fan Chen, and Jonathan P. How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *IROS*, 2018.
- [14] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Soft+hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection. *Neural networks*, 108:466–478, 2018.
- [15] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.
- [16] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE robotics & automation magazine*, 4(1):23–33, 2002.
- [17] Anthony Francis, Claudia Pérez-d’Arpino, Chengshu Li, Fei Xia, Alexandre Alahi, Rachid Alami, Aniket Bera, Abhijat Biswas, Joydeep Biswas, Rohan Chandra, et al. Principles and guidelines for evaluating social robot navigation algorithms. *ACM Transactions on Human-Robot Interaction*, 14(2):1–65, 2025.
- [18] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

- [19] Qihan Gao and He Huang. Evaluation of socially-aware robot navigation: A systematic review and classification. *Frontiers in Robotics and AI*, 8:721317, 2021.
- [20] Jasper Geldenbott and Karen Leung. Legible and proactive robot planning for prosocial human-robot interactions. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13397–13403. IEEE, 2024.
- [21] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. In *Neural Networks*, 2005.
- [22] Silvia Guillen-Ruiz, Juan Pedro Bandera, Alejandro Hidalgo-Paniagua, and Antonio Bandera. Evolution of socially-aware robot navigation. *Electronics*, 12(7):1570, 2023.
- [23] Agrim Gupta, Justin Johnson, Li Fei-Fei Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *CVPR*, 2018.
- [24] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 1995.
- [25] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.
- [26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [27] Yingfan Huang, Huikun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6272–6281, 2019.
- [28] Jarosław Karwowski, Wojciech Szynkiewicz, and Ewa Niewiadomska-Szynkiewicz. Bridging requirements, planning, and evaluation: a review of social robot navigation. *Sensors*, 24(9):2794, 2024.
- [29] Tribhi Kathuria, Ke Liu, Junwoo Jang, X. Jessie Yang, and Maani Ghaffari. Learning implicit social navigation behavior using deep inverse reinforcement learning. *IEEE Robotics and Automation Letters*, 2025.
- [30] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

- [31] Arbaaz Khan, Clark Zhang, Nikolay Atanasov, Konstantinos Karydis, Daniel D. Lee, and Vijay Kumar. End-to-end navigation in unknown environments using neural networks. *arXiv preprint arXiv:1707.07385*, 2017. Workshop on Learning Perception and Control for Autonomous Flight, RSS 2017.
- [32] Beomjoon Kim and Joelle Pineau. Socially adaptive path planning in human environments using inverse reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.
- [33] Bumjun Kim, Mohamed El-Khamy, and Jungwon Lee. Residual lstm: Design of a deep recurrent architecture for distant speech recognition. In *Interspeech*, 2017.
- [34] Jinyeob Kim, Daewon Kwak, Hyunwoo Rim, and Donghan Kim. Belief-aided navigation using bayesian reinforcement learning for avoiding humans in blind spots. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2528–2535. IEEE, 2024.
- [35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [36] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [37] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. In *IJRR*, 2016.
- [38] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.
- [39] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B. Choy, Philip H. S. Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *CVPR*, 2017.
- [40] 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)*, pages 2718–2725. IEEE, 2020.
- [41] NASA Ames Research Center. Nasa task load index (tlx): Paper-and-pencil version. <https://humansystems.arc.nasa.gov/groups/tlx/>, 2022. Accessed: July 14, 2025.

- [42] Brooks Paige, Jan-Willem Van De Meent, Alban Desmaison, Noah Goodman, Pushmeet Kohli, Frank Wood, Philip Torr, et al. Learning disentangled representations with semi-supervised deep generative models. *Advances in neural information processing systems*, 30, 2017.
- [43] Amirreza Payandeh, Daeun Song, Mohammad Nazeri, Jing Liang, Praneel Mukherjee, Amir Hossain Raj, Yangzhe Kong, Dinesh Manocha, and Xuesu Xiao. Social-llava: Enhancing robot navigation through human-language reasoning in social spaces. *arXiv preprint arXiv:2501.09024*, 2024.
- [44] Markus Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *ICRA*, 2017.
- [45] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009.
- [46] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.
- [47] Andrey Rudenko, Laura Palmieri, Michael Herman, and Kai O. Arras. Human motion trajectory prediction: A survey. *International Journal of Robotics Research*, 2019.
- [48] Sebastian Ruder. An overview of multi-task learning in deep neural networks. In *2017 IEEE Transactions on Knowledge and Data Engineering Tutorial*, 2017.
- [49] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajec-tron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. *ECCV*, 2020.
- [50] Phani Teja Singamaneni, Pilar Bachiller-Burgos, Luis J Manso, Anaís Garrell, Alberto Sanfeliu, Anne Spalanzani, and Rachid Alami. A survey on socially aware robot navigation: Taxonomy and future challenges. *The International Journal of Robotics Research*, 43(10):1533–1572, 2024.
- [51] Emrah Akin Sisbot, Luis F. Marin-Urias, Rachid Alami, and Thierry Simeon. A human-aware mobile robot motion planner. In *IEEE Transactions on Robotics*, 2007.

- [52] Daeun Song, Jing Liang, Amirreza Payandeh, Amir Hossain Raj, Xuesu Xiao, and Dinesh Manocha. Vlm-social-nav: Socially aware robot navigation through scoring using vision-language models. *IEEE Robotics and Automation Letters*, 2024.
- [53] Oluwadamilola Sotomi, Devika Kodi, and Aliasghar Arab. Trust through transparency: Explainable social navigation for autonomous mobile robots via vision-language models. *arXiv preprint arXiv:2504.05477*, 2025.
- [54] Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. *Advances in neural information processing systems*, 32, 2019.
- [55] Peter Trautman, Jeremy Ma, Richard M. Murray, and Andreas Krause. Robot navigation in dense human crowds: The case for cooperation. In *ICRA*, pages 2153–2160, 2013.
- [56] Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *Journal of Machine Learning Research*, 15(1), 2014.
- [57] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [58] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. *arXiv preprint arXiv:1710.04689*, 2018.
- [59] Anjiabei Wang and Tesca Fitzgerald. Legibility-aware learning from corrections. 2024.
- [60] 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. IEEE, 2023.
- [61] Greg Welch and Gary Bishop. An introduction to the kalman filter. *UNC Chapel Hill*, 2006. https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf.
- [62] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. In *NIPS Workshop*, 2015.
- [63] Buxin Zeng, Yinghui Pan, Jing Tang, and Yifeng Zeng. Automate legibility through inverse reinforcement learning. *ACM Transactions on Autonomous and Adaptive Systems*, 2025.

- [64] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.