Reinforcement Learning for Autonomous Vehicle Navigation in Factory Environments

Luis Manuel Gutiérrez Lucero
Department of Industrial and Manufacturing Engineering
Universidad Autónoma de Ciudad Juárez
Ciudad Juárez, Chihuahua, México
Email: al173926@alumnos.uacj.mx

***Abstract*—Autonomous navigation in constrained industrial environments presents significant challenges for reinforcement learning due to sparse rewards and safety constraints. This thesis investigates curriculum learning [1] for deep reinforcement learning in factory robot navigation using a PiCar-X platform with multi-modal sensor fusion (camera, ultrasonic, infrared). A navigation framework is developed using Proximal Policy Optimization (PPO) [2] trained in a kinematic simulator with progressive difficulty stages: empty environment (200k timesteps), sparse obstacles (300k timesteps), and full factory layout (500k timesteps), totaling 1M training samples.**

Experimental evaluation in simplified kinematic simulation demonstrates that curriculum learning achieves 22% goal-reaching success where direct training completely fails (0%), with collision-free navigation (0% collision rate in 100 episodes). However, a reward hacking failure mode emerges during extended training (5M timesteps), where the agent maximizes dense shaping rewards (+578) while avoiding goal-reaching, providing insights into reward function design.

Key contributions include: (1) empirical evidence for curriculum learning necessity in constrained navigation, (2) systematic reward hacking analysis with proposed corrections, (3) honest reporting of limitations and failure modes, and (4) an open-source framework for navigation research. Important limitations: All results are from MOCK mode (kinematic simulation without full physics); no hardware validation was performed; only one environment was tested; and sensors are synthetic (procedural gradients, simplified ray-casting). Transfer to physical robots and generalization to other environments remain completely untested and represent significant future work.

***Index Terms*—Reinforcement Learning, Autonomous Navigation, Factory Environments, Mobile Robotics, Proximal Policy Optimization, Robot Operating System, Kinematic Simulation, Curriculum Learning, Multi-Modal Sensing, Industrial Automation.**

I. Introduction

A. Background and Motivation

The integration of autonomous mobile robots into industrial facilities represents a critical evolution in manufacturing automation, with applications spanning material transport, inventory management, and production line coordination [3], [4]. Traditional Automated Guided Vehicles (AGVs) have long served these functions, yet their reliance on fixed infrastructure—magnetic tapes, inductive wires, or precisely mapped environments—limits their flexibility and adaptability to dynamic operational conditions [5]. As modern manufacturing paradigms shift toward agile, reconfigurable production systems, there emerges a compelling need for autonomous navigation solutions capable of operating in unstructured, dynamic environments without extensive environmental modification.

Classical robotic navigation approaches typically decompose the problem into perception, localization, mapping, path planning, and control modules [5]. While Simultaneous Localization and Mapping (SLAM) combined with global planners (e.g., A* [6], D* [7], RRT [8]) has achieved remarkable success in structured environments, these methods face fundamental limitations when confronted with:

1) **Dynamic Obstacles:** Moving personnel, forklifts, and temporary equipment placements require continuous replanning, which can be computationally prohibitive for real-time operation.
2) **Computational Constraints:** Factory robots often operate on embedded platforms with limited processing power, precluding sophisticated SLAM and planning algorithms.
3) **Sensor Limitations:** Cost constraints and operational requirements may restrict the use of expensive sensors such as 3D LiDAR, necessitating reliance on vision, ultrasonic, and simple infrared sensors.
4) **Environmental Variation:** Factory lighting conditions, floor reflectivity, and temporary layout changes introduce perceptual uncertainty that challenges traditional handcrafted feature extraction pipelines.

Deep Reinforcement Learning (DRL) offers a paradigm shift for robotic navigation by enabling end-to-end learning of control policies directly from sensory observations without explicit feature engineering or hand-coded planning heuristics [9]. Through iterative interaction with the environment, DRL agents can discover adaptive behaviors that implicitly encode obstacle avoidance, goal-directed navigation, and dynamic interaction strategies. Recent successes in DRL for navigation—from DeepMind's DQN mastering Atari games [10] to advanced continuous control with Proximal Policy Optimization (PPO) [2]—demonstrate the potential of learned policies to outperform traditional methods in complex, partially observable environments [11].

This thesis focuses on the SunFounder PiCar-X, an Ackermann-steering robotic platform equipped with a Raspberry Pi 4 single-board computer. The PiCar-X's sensor suite includes a wide-angle camera for visual perception, an ultrasonic range finder for proximal obstacle detection, and a five-channel infrared line-tracking array—representative of low-cost industrial sensing configurations. The platform's computational constraints (Raspberry Pi 4 with 4GB RAM) and Ackermann steering kinematics mirror the operational realities of affordable factory automation solutions.

B. Problem Statement

This thesis addresses the following central research question:

Can a deep reinforcement learning agent, trained in kinematic simulation, learn collision-free navigation behaviors in factory-like environments using low-cost, multi-modal sensors, and what training strategies (e.g., curriculum learning) are necessary to enable learning in this challenging domain?

This question encompasses several technical challenges:

1) **State Representation:** Designing a compact yet informative observation space that captures relevant environmental features from heterogeneous sensors (camera, ultrasonic, infrared) while remaining tractable for policy learning.
2) **Reward Engineering:** Formulating a reward function that balances multiple objectives—goal-reaching efficiency, collision avoidance, path smoothness, and motion quality—without introducing spurious local optima or reward hacking.
3) **Simulation Fidelity:** Understanding the gap between kinematic simulation and full physics simulation, with domain randomization [12] designed as future work for eventual physical deployment.
4) **Sample Efficiency:** Training a performant policy within reasonable computational budgets, given that on-policy algorithms like PPO require substantial environment interaction.
5) **Safety and Reliability:** Ensuring collision avoidance and graceful degradation in the presence of sensor noise, partial occlusions, or unexpected obstacles.

C. Research Objectives

1) General Objective: To develop, implement, and evaluate a deep reinforcement learning framework for autonomous navigation of a mobile robot in simulated factory environments, investigating curriculum learning effectiveness for enabling policy learning in constrained spaces with sparse rewards, and establishing a foundation for future simulation-to-reality transfer research.

2) Specific Objectives:

1) **Problem Formulation:** Mathematically formulate the factory navigation task as a Partially Observable Markov Decision Process (POMDP) [11], specifying observation space, action space, transition dynamics, and reward structure.
2) **Algorithm Selection and Implementation:** Select and implement Proximal Policy Optimization (PPO) with appropriate neural network architectures for multi-modal sensor fusion and continuous control.
3) **Simulation Environment Development:** Design and implement a high-fidelity Gazebo [13] simulation environment representing factory layouts with static and dynamic obstacles, integrated with ROS 2 [14] for modular sensor and actuator interfaces.
4) **Multi-Modal Sensor Integration:** Develop observation preprocessing pipelines for camera images, ultrasonic range data, and infrared line sensors, including normalization, frame stacking, and dimensionality reduction.
5) **Reward Function Design and Iterative Tuning:** Engineer a shaped reward function that guides policy learning toward safe, efficient, and smooth navigation behaviors, then systematically tune reward coefficients based on observed policy behavior to improve goal-directed navigation.
6) **Training Protocol and Extended Experimentation:** Conduct baseline training (500k timesteps) followed by extended training (1M timesteps) with reward function refinements, documenting the impact of iterative development on policy performance. All training was conducted exclusively in MOCK mode, a kinematic simulation without full Gazebo physics integration, enabling rapid algorithm iteration. See Section VII for a discussion of this limitation and future integration plans.
7) **Evaluation Methodology:** Establish evaluation metrics (success rate, collision rate, reward, distance to goal, policy variance) and experimental protocols, including comparative analysis between training iterations and random baseline.
8) **Performance Analysis and Hypothesis Testing:** Systematically investigate training-evaluation discrepancies, policy convergence behavior, and the effectiveness of reward engineering through quantitative analysis and hypothesis-driven experimentation.

Note that domain randomization strategies, classical baseline comparisons, and sim-to-real deployment were designed as part of the framework but remain future work beyond the scope of the current experimental validation.

D. Contributions

This thesis makes the following contributions to the field of autonomous mobile robotics.
textbfNote: All results are from simplified kinematic simulation (MOCK mode); hardware validation remains future work:

1) **Comprehensive DRL Framework with Iterative Development Methodology:** A reproducible framework for learning navigation policies in factory environments, demonstrated through two training iterations (500k and 1M timesteps) that illustrate iterative reward engineering principles and policy refinement strategies.
2) **Multi-Modal Policy Architecture:** Implementation of stable-baselines3's `MultiInputPolicy` to effectively fuse heterogeneous sensor modalities (camera, ultrasonic, infrared) for continuous-control navigation, achieving perfect collision avoidance (0% collision rate) in cluttered environments.

3) **Empirical Validation of Reward Engineering:** Quantitative demonstration that reward coefficient tuning ($\lambda_{\text{progress}}$: $2.0 \to 3.5$) produces measurable behavioral changes (16% reward improvement, 20% reduction in distance-to-goal, 62% variance reduction), providing empirical evidence for reward shaping effectiveness.
4) **Training-Evaluation Discrepancy Analysis:** Systematic investigation of policy performance differences between stochastic training and deterministic evaluation, identifying critical sensitivities to sampling mode, environment fidelity, and value function convergence—advancing understanding of RL evaluation methodology.
5) **Open-Source Implementation:** A fully documented ROS 2 workspace with complete training scripts, evaluation protocols, and visualization tools, enabling direct reproducibility and facilitating future research on factory navigation and reward engineering methodologies.

E. Thesis Organization

The remainder of this thesis is organized as follows:

- **Section II** reviews theoretical foundations of reinforcement learning, surveys related work on DRL-based navigation, discusses simulation-to-reality transfer considerations, and examines prior applications in industrial environments.
- **Section III** presents the mathematical problem formulation, details the PPO algorithm, describes the multi-modal observation space and continuous action space, and derives the reward function.
- **Section IV** describes the system architecture, ROS 2 package structure, Gazebo simulation setup, robot model configuration, sensor implementations, and domain randomization techniques.
- **Section V** outlines the experimental design, defines evaluation metrics, describes training procedures, and specifies baseline comparison methods.
- **Section VI** presents comprehensive experimental results from curriculum learning experiments, including learning curves, performance comparisons across training paradigms, and detailed analysis of the reward hacking failure mode.
- **Section VII** discusses implications of findings, compares with traditional navigation methods, addresses limitations, and explores future research directions.
- **Section VIII** summarizes key contributions, reiterates findings, and outlines paths for future work.

F. Scope and Limitations

This research scope and its limitations are described below.

Completed Work:

- Development and training of DRL policies using kinematic simulation (MOCK mode) with factory-representative environments
- Multi-modal sensor integration (camera, ultrasonic, infrared line tracking)
- PPO algorithm implementation using stable-baselines3
- Comprehensive evaluation against random baseline across multiple training paradigms
- Curriculum learning framework with three-stage progressive difficulty

Designed but Not Implemented (Future Work):

- Full Gazebo physics integration via ROS 2 topics
- Domain randomization strategies (designed but not empirically validated)
- Deployment on Raspberry Pi 4 hardware
- Classical baseline comparisons (A*, DWA)
- Sensor modality ablation studies

Out of Scope:

- Multi-agent coordination and fleet management
- Long-term autonomy and lifelong learning
- Safety certification for industrial deployment
- Alternative DRL algorithms beyond PPO
- Human-robot interaction and collaborative navigation

Key Limitations (Important for Interpreting Results):

1) **Kinematic Simulation Only:** All training and evaluation used MOCK mode—a simplified kinematic simulation without full physics. The environment does not model friction, inertia, motor dynamics, realistic sensor noise, or visual rendering artifacts.
2) **No Hardware Validation:** The trained policy has not been deployed to physical hardware. Sim-to-real transfer remains completely untested and would likely require significant domain adaptation.
3) **Single Environment:** Only one factory layout was tested. Generalization to different environments, obstacle configurations, or lighting conditions was not evaluated.
4) **Synthetic Sensors:** Camera observations are procedurally generated gradients (not realistic images) and lidar is simplified ray-casting without noise or occlusion artifacts.
5) **Limited Baselines:** Only random policy baseline was tested. No comparison with classical navigation (DWA, A*) or other RL algorithms (SAC, TD3).

These limitations significantly constrain the generalizability of findings and must be considered when interpreting all results presented in this thesis.

II. Theoretical Foundation and Related Work

A. Reinforcement Learning: Mathematical Framework

Reinforcement Learning (RL) provides a mathematical framework for sequential decision-making under uncertainty, wherein an agent learns to maximize cumulative reward through interaction with an environment [15]. The navigation problem is formalized as a Partially Observable Markov Decision Process (POMDP) [11], defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \gamma)$, where:

- $\mathcal{S}$ is the state space, representing the true (unobserved) environment state
- $\mathcal{A}$ is the action space
- $\mathcal{O}$ is the observation space, the sensory information available to the agent
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the state transition probability function, where $\Delta(\mathcal{S})$ denotes the probability simplex over $\mathcal{S}$

- $\mathcal{R}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function
- $\mathcal{Z}: \mathcal{S} \to \Delta(\mathcal{O})$ is the observation function mapping states to observation distributions
- $\gamma \in [0, 1)$ is the discount factor controlling the present value of future rewards

At each discrete timestep t, the agent receives observation $o_t \in \mathcal{O}$, selects action $a_t \in \mathcal{A}$ according to its policy $\pi(a_t|o_t)$, transitions to a new state $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$, and receives scalar reward $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$. The agent's objective is to learn a policy π^* that maximizes the expected discounted cumulative return:

$$J(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] = \mathbb{E}_{\tau \sim \pi}[G_0] \tag{1}$$

where $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ denotes a trajectory sampled by executing policy π, and $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the return from timestep t.

The **value function** $V^\pi(s)$ quantifies the expected return starting from state s and following policy π:

$$V^\pi(s) = \mathbb{E}_\pi\left[G_t \mid s_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s\right] \tag{2}$$

The **action-value function** (Q-function) $Q^\pi(s, a)$ represents the expected return from taking action a in state s and subsequently following π:

$$Q^\pi(s, a) = \mathbb{E}_\pi\left[G_t \mid s_t = s, a_t = a\right] \tag{3}$$

The **advantage function** measures the relative benefit of action a over the average action in state s:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \tag{4}$$

The optimal policy π^* satisfies $V^{\pi^*}(s) \geq V^\pi(s)$ for all $s \in \mathcal{S}$ and all policies π. These optimal value functions satisfy the **Bellman optimality equations**:

$$V^*(s) = \max_{a \in \mathcal{A}} \left\{\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)}[V^*(s')]\right\} \tag{5}$$

$$Q^*(s, a) = \mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s,a)}\left[\max_{a'} Q^*(s', a')\right] \tag{6}$$

B. Deep Reinforcement Learning Algorithms

Classical RL methods become intractable in high-dimensional state and action spaces. Deep Reinforcement Learning (DRL) addresses this limitation by using deep neural networks as function approximators for policies and value functions, enabling learning from raw sensory inputs such as images [10].

1) Value-Based Methods: Deep Q-Networks (DQN): Value-based methods learn an approximation $Q(s, a; \theta)$ of the optimal action-value function using neural networks with parameters θ. Deep Q-Networks (DQN) [10] introduced two critical innovations: *experience replay* [16] and *target networks*. The DQN loss function is:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right] \tag{7}$$

where $\mathcal{D}$ is the replay buffer and θ^- are the target network parameters updated periodically from θ. However, DQN is designed for discrete action spaces and requires modifications (e.g., discretization) for continuous control, introducing approximation errors.

2) Policy Gradient Methods: Policy gradient methods directly parameterize the policy $\pi(a|s; \theta)$ and optimize it by ascending the gradient of expected return $J(\theta)$. The policy gradient theorem [15], [17] states:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi(a_t|s_t; \theta)\, A^{\pi_\theta}(s_t, a_t)\right] \tag{8}$$

This gradient is estimated via Monte Carlo sampling from trajectories. The REINFORCE algorithm uses this estimate directly, but suffers from high variance. Actor-Critic methods reduce variance by using a learned value function $V(s; \phi)$ as a baseline to estimate advantages:

$$\hat{A}_t = r_t + \gamma V(s_{t+1}; \phi) - V(s_t; \phi) \tag{9}$$

3) Proximal Policy Optimization (PPO): Proximal Policy Optimization (PPO) [2] is a policy gradient method that has emerged as the de facto standard for continuous control in robotics due to its simplicity, stability, and strong empirical performance. PPO addresses the challenge of determining appropriate step sizes in policy updates: too small leads to slow learning, too large causes performance collapse.

PPO introduces a *clipped surrogate objective* that constrains policy updates to a trust region. Let $r_t(\theta) = \frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_{\text{old}})}$ denote the probability ratio between the new and old policies. The PPO-Clip objective is:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t,\, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t\right)\right] \tag{10}$$

where $\epsilon \in (0, 1)$ (commonly $\epsilon = 0.2$) controls the clipping range, and $\hat{A}_t$ is the advantage estimate at timestep t. The clipping operation prevents the probability ratio from deviating too far from 1, effectively constraining the KL divergence between consecutive policies.

The complete PPO objective includes a value function loss and an entropy bonus:

$$\mathcal{L}^{\text{PPO}}(\theta, \phi) = \hat{\mathbb{E}}_t\left[\mathcal{L}_t^{\text{CLIP}}(\theta) - c_1 \mathcal{L}_t^{\text{VF}}(\phi) + c_2 S[\pi_\theta](s_t)\right] \tag{11}$$

where:

- $\mathcal{L}_t^{\text{VF}}(\phi) = (V(s_t; \phi) - V_t^{\text{target}})^2$ is the value function loss, where V_t^{target} is computed using Generalized Advantage Estimation (GAE) returns: $V_t^{\text{target}} = \hat{A}_t^{\text{GAE}} + V(s_t; \phi_{\text{old}})$
- $S[\pi_\theta](s_t) = -\sum_a \pi(a|s_t; \theta) \log \pi(a|s_t; \theta)$ is the policy entropy encouraging exploration[1]
- $c_1, c_2 > 0$ are coefficients balancing the three terms (typically $c_1 = 0.5$, $c_2 = 0.01$)

[1]For continuous action spaces with Gaussian policies $\pi(a|s) = \mathcal{N}(\mu(s), \sigma^2 I)$, the entropy has the closed-form expression $S[\pi](s) = \frac{1}{2}\sum_{i=1}^{|\mathcal{A}|} \log(2\pi e \sigma_i^2)$, which depends only on the policy standard deviation.

Algorithm 1 Proximal Policy Optimization (PPO-Clip)

```
 1: Initialize policy $\theta_0$ and value function $\phi_0$
 2: for iteration $k = 0, 1, 2, \dots$ do
 3:     Collect trajectories $\{\tau_i\}$ by $\pi_{\theta_k}$
 4:     Compute advantages $\hat{A}_t$ using GAE
 5:     for epoch $e = 1, \dots, E$ do
 6:         for minibatch $(s, a, \hat{A}, V^{\text{tgt}})$ do
 7:             Update $\theta$ maximizing $\mathcal{L}^{\text{CLIP}}(\theta)$
 8:             Update $\phi$ minimizing $\mathcal{L}^{\text{VF}}(\phi)$
 9:         end for
10:     end for
11:     $\theta_{k+1} \leftarrow \theta$, $\phi_{k+1} \leftarrow \phi$
12: end for
```

Advantages are typically estimated using Generalized Advantage Estimation (GAE) [18], which provides a variance-reduced estimator:

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \tag{12}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the temporal-difference error, and $\lambda \in [0, 1]$ is the GAE parameter trading off bias and variance ($\lambda = 1$ recovers Monte Carlo returns, $\lambda = 0$ gives one-step TD) [18].

PPO Algorithm Summary:

Rationale for PPO Selection: For this thesis, PPO is selected over alternative DRL algorithms for the following reasons:

1) **Continuous Control:** The PiCar-X requires continuous steering and throttle commands; PPO naturally handles continuous action spaces via Gaussian policies.
2) **Sample Efficiency vs. Stability:** While off-policy algorithms like SAC may be more sample-efficient, PPO offers superior training stability—critical for robotics where policy failures can lead to unsafe behavior.
3) **Proven Robotics Performance:** PPO has been reported in numerous robotic manipulation and locomotion tasks, as reported in prior literature.
4) **Implementation Maturity:** The stable-baselines3 library provides a well-tested PPO implementation with comprehensive documentation and community support.

4) Alternative Algorithms: SAC and TD3: Soft Actor-Critic (SAC) [19] is an off-policy algorithm maximizing both expected return and policy entropy, promoting exploration and robustness. Twin Delayed Deep Deterministic Policy Gradient (TD3) [20] addresses overestimation bias in Q-learning via twin critics and delayed policy updates. While these algorithms offer higher sample efficiency, their increased complexity and potential instability make PPO the preferred choice for initial development. If time permits, SAC will be evaluated as a secondary baseline.

C. Sim-to-Real Transfer and Domain Randomization

A central challenge in learned robotics is the *sim-to-real gap*—the discrepancy between simulation and reality that causes policies trained in simulation to perform poorly on physical systems [21]. Sources of this gap include:

- **Physics Modeling Errors:** Simplified contact dynamics, friction models, and actuation delays
- **Sensor Discrepancies:** Noise characteristics, latency, field-of-view differences
- **Visual Appearance:** Texture quality, lighting conditions, material reflectance
- **Unmodeled Dynamics:** Wheel slip, actuator backlash, battery voltage variations

Domain Randomization [21] is a powerful technique to bridge this gap by training policies over a distribution of simulation parameters, training the policy to be more tolerant of variations that encompass real-world conditions. Key randomization strategies include:

1) **Visual Randomization:**
    - Lighting intensity and color temperature
    - Object textures and colors
    - Camera position jitter and field-of-view variation
    - Ambient occlusion and shadow rendering parameters
2) **Dynamics Randomization:**
    - Wheel friction coefficients
    - Robot mass and inertia
    - Actuator gain and latency
    - Ground surface properties (friction, compliance)
3) **Sensor Noise Injection:**
    - Gaussian noise on ultrasonic range measurements
    - Image noise (Gaussian, salt-and-pepper, motion blur)
    - Infrared sensor threshold variation
4) **Obstacle and Layout Randomization:**
    - Starting position and goal location
    - Static obstacle placement
    - Dynamic obstacle trajectories and speeds

Formally, let ξ denote the vector of simulation parameters. Instead of training on a single fixed ξ_0, $\xi \sim p_\Xi(\xi)$ is sampled at the start of each episode, where p_Ξ is a randomization distribution designed to cover the expected real-world parameter space. The policy objective becomes:

$$\pi^* = \arg\max_\pi \mathbb{E}_{\xi \sim p_\Xi} [J(\pi; \xi)] \tag{13}$$

The hypothesis underlying domain randomization is that a policy robust to a wide range of simulated conditions will generalize to the specific (but unknown) real-world conditions, provided the real-world parameters lie within or near the randomization distribution's support.

D. Mobile Robot Navigation: Classical Approaches

Before discussing RL-based navigation, classical approaches are briefly reviewed to contextualize the contributions of this work.

1) Simultaneous Localization and Mapping (SLAM): SLAM algorithms [5], [22] enable robots to construct maps of unknown environments while simultaneously localizing within those maps. State-of-the-art SLAM systems (ORB-SLAM, Cartographer) achieve centimeter-level accuracy using LiDAR or cameras. However, SLAM requires:

- Significant computational resources (often GPU acceleration)
- Static environment assumptions or explicit dynamic object filtering
- Loop closure detection for long-term consistency
- Calibrated sensors with known extrinsics

These requirements are often prohibitive for low-cost factory robots operating in dynamic environments.

2) Global Path Planning: Given a map, global planners (A*, D*, RRT*, etc.) compute optimal or near-optimal paths from start to goal. However:

- They assume map accuracy and do not adapt to dynamic changes without replanning
- Computational cost scales with map resolution and environment complexity
- They produce geometric paths that require separate trajectory tracking controllers

3) Local Reactive Control: Local methods such as Dynamic Window Approach (DWA) [23] generate control commands directly from sensor readings without explicit maps. While computationally efficient and reactive to obstacles, they suffer from local minima, oscillations in tight spaces, and lack of global path optimality.

E. Deep Reinforcement Learning for Navigation: Literature Review

Recent years have witnessed an explosion of research applying DRL to mobile robot navigation. This work is categorized by problem domain, and key contributions are highlighted.

1) General Indoor Navigation: Zhu et al. (2023) provide a comprehensive review of DRL for mobile robot navigation in dynamic environments [4], identifying three main approaches:

1) **End-to-End Learning:** Policies map raw sensor inputs (images, LiDAR) directly to motor commands, bypassing traditional perception pipelines.
2) **Hierarchical Learning:** High-level policies select subgoals or behaviors; low-level policies execute primitive skills.
3) **Hybrid Methods:** Combine learned components with classical modules (e.g., learned perception with model-based planning).

Key challenges identified include sparse rewards, partial observability, sample inefficiency, and sim-to-real transfer—which this thesis investigates through curriculum learning and reward engineering.

2) Warehouse and Factory Navigation: Peyas et al. (2022) applied Deep Q-Learning to warehouse robot navigation [3], demonstrating that DRL agents can learn collision-free navigation in gridworld environments. However, their work used discrete action spaces and simplified 2D observations, limiting applicability to real robots with continuous control and high-dimensional sensing.

Frimodig (2024) [24] employed PPO for multi-AGV routing on factory floors, optimizing routes for multiple vehicles to improve throughput over heuristic dispatching rules. This work focused on fleet coordination rather than individual navigation control, treating path execution as deterministic.

The present work differs by focusing on single-robot, continuous-control navigation with realistic multi-modal sensing, addressing obstacle avoidance and goal-directed navigation in simulation.

3) Visual Navigation: Recent work on visual navigation has demonstrated end-to-end learning from RGB images. Notable examples include:

- Learning to navigate office environments from 84×84 RGB images using PPO with convolutional neural networks
- PointGoal navigation in photorealistic environments (Habitat, Gibson)
- Vision-based drone racing using DRL with aggressive maneuvers

These works validate the feasibility of learning navigation policies from high-dimensional visual inputs. However, factory environments introduce additional constraints (line following, ultrasonic integration) not addressed in prior work.

4) Multi-Modal Sensor Fusion: Few studies have addressed multi-modal sensor fusion in DRL navigation. Most work uses either vision alone or LiDAR alone. The contribution of this work—fusing camera, ultrasonic, and infrared sensors within a single policy architecture—addresses a gap in the literature, particularly for low-cost robotic platforms.

F. Gymnasium and Stable-Baselines3

Gymnasium [25], [26] (formerly OpenAI Gym) is the standard Python API for RL environments, defining interfaces for `reset()`, `step(action)`, and `render()`. Custom environments are implemented by subclassing `gymnasium.Env` and defining:

- `observation_space`: A `gym.Space` object (Box, Dict, etc.)
- `action_space`: A `gym.Space` object
- `reset()`: Returns initial observation o_0
- `step(action)`: Returns $(o_{t+1}, r_t, \text{terminated}, \text{truncated}, \text{info})$

Stable-Baselines3 [27] (SB3) is a high-quality implementation of DRL algorithms in PyTorch, providing:

- PPO, A2C, SAC, TD3, DQN implementations with unified interfaces
- Vectorized environments for parallel rollouts
- Policy architectures: MlpPolicy, CnnPolicy, MultiInputPolicy
- Tensorboard logging, model checkpointing, hyperparameter optimization

For the multi-modal observation space (camera, ultrasonic, IR), `MultiInputPolicy` was employed, which accepts a `Dict` observation space and independently processes each modality through specialized feature extractors before fusing features in a shared MLP policy head.

III. Methodology: Problem Formulation and Algorithm Design

This section presents the formal problem formulation as a POMDP, defines the observation and action spaces, derives the reward function, and specifies the neural network architecture and training procedures.

A. POMDP Formulation for Factory Navigation

The factory navigation task is modeled as a POMDP [11] $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \gamma)$ with the following instantiation:

1) State Space $\mathcal{S}$: The true state $s_t \in \mathcal{S}$ encompasses all environment information, including:

- Robot pose: position $(x, y) \in \mathbb{R}^2$, orientation $\theta \in [0, 2\pi)$
- Robot velocities: linear $v \in \mathbb{R}$, angular $\omega \in \mathbb{R}$
- Goal location: $(x_g, y_g) \in \mathbb{R}^2$
- Obstacle positions and velocities (static and dynamic)
- Line track geometry (if present)

The state space is continuous and high-dimensional. Crucially, the agent does not have direct access to s_t (no ground-truth localization).

2) Action Space $\mathcal{A}$: The action space is continuous, representing commands to the robot's actuators. The action space is defined as:

$$a_t = (\text{throttle}, \text{steering}) \in \mathcal{A} = [0.0, 1.0] \times [-1.0, 1.0] \quad (14)$$

where:

- **Throttle** $\in [0.0, 1.0]$: Forward speed command (normalized), corresponding to $0\,\mathrm{m\,s^{-1}}$ to $1.0\,\mathrm{m\,s^{-1}}$ on the physical PiCar-X.
- **Steering** $\in [-1.0, 1.0]$: Angular velocity or steering angle (normalized). Negative values correspond to left turns, positive to right turns.

These actions are published to the ROS 2 topic `/picarx/cmd_vel` as `geometry_msgs/Twist` messages:

$$\text{linear.x} = \text{throttle} \times v_{\max} \quad (15)$$

$$\text{angular.z} = \text{steering} \times \omega_{\max} \quad (16)$$

where $v_{\max} = 1.0\,\mathrm{m\,s^{-1}}$ and $\omega_{\max} = 1.5\,\mathrm{rad\,s^{-1}}$ are scaling factors.

3) Observation Space $\mathcal{O}$: The observation $o_t \in \mathcal{O}$ consists of multi-modal sensor data available to the robot:

$$o_t = \{I_t, U_t, L_t\} \quad (17)$$

where:

1) **Camera Image** $I_t \in [0, 255]^{H \times W \times 3}$:
   - RGB image from front-facing camera
   - Native capture resolution of 160×120 pixels, downsampled to $H = 60, W = 80$ pixels for the observation space
   - Normalized to $[0, 1]$ by dividing by 255
   - Provides visual context for obstacle detection, lane identification, and goal recognition
2) **Lidar Range** $U_t \in [0.0, 2.5]$:
   - 61-beam lidar with $\pm 25.8°$ field of view ($0.86\,\mathrm{rad}$ total)
   - Range: $0.03\,\mathrm{m}$ to $2.5\,\mathrm{m}$
   - Processed into a single minimum-distance value for the observation space, providing explicit distance information for collision avoidance
   - Published on `/picarx/scan` as `sensor_msgs/LaserScan`
3) **Infrared Line Sensors** $L_t \in \{0, 1\}^3$:
   - Three-channel binary sensor array detecting line contrast
   - Channels: [left, center, right]
   - Binary values: 1 = line detected, 0 = no line
   - Enables line-following behavior when lane markings are present
   - Published on custom topic `/picarx/line_sensors`
   - **Note:** IR sensors are simulated in MOCK mode but not physically modeled in the Gazebo SDF; Gazebo integration would require custom plugin development

The observation space is formalized as a `gymnasium.spaces.Dict`:

```
observation_space = Dict({
  'image': Box(low=0, high=1,
    shape=(60,80,3), dtype=np.float32),
  'lidar': Box(low=0, high=2.5,
    shape=(1,), dtype=np.float32),
  'line_sensors': MultiBinary(3)
})
```

This multi-modal observation space is particularly well-suited for factory navigation:

- **Camera** captures global scene context, recognizes goals, and detects distant obstacles
- **Lidar** (processed to minimum distance) provides accurate near-range distance for reactive collision avoidance
- **Line sensors** enable lane-keeping on marked paths, complementing vision-based navigation

4) Transition Dynamics $\mathcal{P}$: The transition function $\mathcal{P}(s_{t+1}|s_t, a_t)$ is determined by:

- **Robot Kinematics:** Differential-drive or Ackermann steering model
- **Physics Simulation:** Gazebo's ODE physics engine models contact, friction, and inertia
- **Dynamic Obstacles:** Scripted or learned behaviors for moving objects

The transition is stochastic due to sensor noise, wheel slip, and dynamic obstacle behavior.

5) Observation Function $\mathcal{Z}$: The observation function $\mathcal{Z}(o_t|s_t)$ maps true states to sensor observations. In simulation, this is implemented via Gazebo sensor plugins. In reality, sensor noise and latency introduce additional stochasticity.

6) Reward Function $\mathcal{R}$: The reward function is the primary mechanism for shaping learned behavior. Following established principles in reward shaping [28], a multi-term reward is designed to balance goal-reaching, collision avoidance, path efficiency, and motion smoothness:

$$r_t = r_t^{\text{goal}} + r_t^{\text{collision}} + r_t^{\text{potential}} + r_t^{\text{heading}} + r_t^{\text{velocity}} + r_t^{\text{smooth}} + r_t^{\text{proximity}} \quad (18)$$

1. Goal Reaching Reward r_t^{goal}:

$$r_t^{\text{goal}} = \begin{cases} +200 & \text{if } d_{\text{goal}}(s_t) \leq r_{\text{thresh}} \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

where $d_{\text{goal}}(s_t) = \|(x_t, y_t) - (x_g, y_g)\|_2$ is the Euclidean distance to goal, and $r_{\text{thresh}} = 0.5\,\text{m}$ is the goal radius. This sparse reward provides a strong learning signal upon task completion.

2. Collision Penalty $r_t^{\text{collision}}$:

$$r_t^{\text{collision}} = \begin{cases} -100 & \text{if } d_{\text{obs}}(s_t) < d_{\text{coll}} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

where $d_{\text{obs}}(s_t)$ denotes the minimum distance to the nearest obstacle and $d_{\text{coll}} = 0.15\,\text{m}$ is the collision threshold. This penalty terminates the episode and strongly discourages unsafe behavior. The magnitude of -100 was chosen to be substantial relative to typical episode rewards while providing a strong safety-first signal to the policy. Note that the proximity shaping reward (below) uses a separate threshold of $d_{\text{safe}} = 0.3\,\text{m}$ to provide early warning before actual collisions.

3. Potential-Based Shaping $r_t^{\text{potential}}$: To provide dense learning signal while preserving optimal policy invariance, we employ potential-based reward shaping as formalized by Ng et al. [28]. Define the potential function as:

$$\Phi(s) = -\lambda_{\text{pot}} \cdot d_{\text{goal}}(s) \tag{21}$$

The shaping reward is then the difference in discounted potentials:

$$r_t^{\text{potential}} = \gamma\Phi(s_t) - \Phi(s_{t-1}) = \lambda_{\text{pot}} \cdot [d_{\text{goal}}(s_{t-1}) - \gamma \cdot d_{\text{goal}}(s_t)] \tag{22}$$

with $\lambda_{\text{pot}} = 10.0$ and discount factor $\gamma = 0.99$. This formulation ensures that cumulative shaping rewards telescope, eliminating concerns about reward hacking through extended episodes [28]. The potential-based structure guarantees that any policy optimal under the shaped reward is also optimal under the original sparse reward.

4. Heading Alignment Reward r_t^{heading}: To encourage the agent to orient toward the goal while moving, a heading alignment term rewards motion in the correct direction:

$$r_t^{\text{heading}} = \lambda_{\text{head}} \cdot \cos(\theta_{\text{err}}) \cdot v_t \tag{23}$$

where $\theta_{\text{err}} = \theta_t - \arctan 2(y_g - y_t, x_g - x_t)$ is the angular error between the robot's heading θ_t and the bearing to goal, and v_t is the current linear velocity. The coefficient $\lambda_{\text{head}} = 1.0$ weights this term. This reward is maximized when the robot moves directly toward the goal ($\cos(\theta_{\text{err}}) = 1$) at high velocity, and provides zero reward when stationary regardless of orientation.

5. Velocity Bonus r_t^{velocity}: To discourage overly conservative behavior and encourage efficient navigation:

$$r_t^{\text{velocity}} = \lambda_{\text{vel}} \cdot v_t \tag{24}$$

with $\lambda_{\text{vel}} = 0.5$. This modest bonus rewards forward progress and helps prevent the policy from learning to remain stationary to avoid collision penalties.

6. Smoothness Penalty r_t^{smooth}: Large changes in control commands are penalized to promote smooth driving and reduce actuator wear:

$$r_t^{\text{smooth}} = -\lambda_{\text{smooth}} \cdot \|a_t - a_{t-1}\|_2 \tag{25}$$

with $\lambda_{\text{smooth}} = 0.05$. This reduces jerk and oscillatory behavior, producing trajectories more suitable for physical deployment.

7. Proximity Shaping $r_t^{\text{proximity}}$: To encourage maintaining safe clearance from obstacles before collision becomes imminent:

$$r_t^{\text{proximity}} = \begin{cases} -\lambda_{\text{prox}} \cdot (d_{\text{safe}} - d_{\text{obs}}(s_t)) & \text{if } d_{\text{obs}}(s_t) < d_{\text{safe}} \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

with $\lambda_{\text{prox}} = 1.0$ and $d_{\text{safe}} = 0.3\,\text{m}$. This provides a continuous penalty that increases as the robot approaches obstacles, serving as an early warning mechanism before the discrete collision penalty applies.

Rationale: This reward structure balances multiple objectives through a principled hierarchy:

- **Sparse terminal rewards** (+200 goal, -100 collision) provide clear success/failure signals that define the task objective
- **Potential-based shaping** (Eq. 22) guides exploration toward goal states while theoretically preserving optimal policy invariance [28]
- **Auxiliary rewards** (heading alignment, velocity bonus) encourage desirable motion characteristics without dominating the primary objective
- **Regularization terms** (smoothness, proximity) improve trajectory quality and safety margins

The potential-based formulation addresses a critical issue in reward design: with traditional progress rewards $r = \lambda(d_{t-1} - d_t)$, accumulated shaping can exceed terminal bonuses for long episodes, incentivizing timeout strategies. The potential-based structure ensures that $\sum_{t=0}^{T} r_t^{\text{potential}} = \lambda_{\text{pot}}[d_{\text{goal}}(s_0) - \gamma^T d_{\text{goal}}(s_T)]$, which is bounded and does not grow with episode length.

7) Discount Factor γ and Episode Configuration: The discount factor is set to $\gamma = 0.99$, balancing short-term reactivity with long-term planning. The maximum episode length was configured as $T_{\max} = 500$ steps (50 seconds at 10 Hz) for the baseline 500k training, then extended to $T_{\max} = 750$ steps (75 seconds) for the extended 1M training, and $T_{\max} = 1000$ steps for the curriculum model to provide additional time for goal-reaching. With $\gamma = 0.99$, the effective planning horizon spans approximately 100 steps, ensuring that the agent considers long-term consequences of actions.

Note on Episode Lengths: The $T_{\max}$ values (500, 750, 1000) represent *maximum allowed steps* per episode—the upper bound before timeout termination. Observed mean episode lengths (e.g., 837 steps in Table IV) reflect actual episode durations, which may be shorter due to goal-reaching or collision, or approach $T_{\max}$ due to timeout.

B. Neural Network Policy Architecture

The policy $\pi(a|o;\theta)$ is implemented as a neural network with architecture tailored for multi-modal input fusion. Stable-baselines3's `MultiInputPolicy` is employed, which processes each observation modality through specialized feature extractors.

1) Feature Extractors: **1. Convolutional Neural Network (CNN) for Camera Images:** The camera image $I_t \in \mathbb{R}^{60\times 80\times 3}$ is processed by a CNN feature extractor:

$$h_1 = \text{ReLU}(\text{Conv2D}(I_t, 32 \text{ filters}, 8\times 8, \text{stride } 4)) \quad (27)$$
$$h_2 = \text{ReLU}(\text{Conv2D}(h_1, 64 \text{ filters}, 4\times 4, \text{stride } 2)) \quad (28)$$
$$h_3 = \text{ReLU}(\text{Conv2D}(h_2, 64 \text{ filters}, 3\times 3, \text{stride } 1)) \quad (29)$$
$$z_{\text{img}} = \text{Flatten}(h_3) \in \mathbb{R}^{d_{\text{img}}} \quad (30)$$

This is the standard Nature DQN architecture adapted for the specified image dimensions. For input dimensions $H = 60$, $W = 80$, the spatial dimensions evolve as follows:

- After Conv1 (8×8, stride 4): $\lfloor(60-8)/4\rfloor + 1 = 14$, $\lfloor(80-8)/4\rfloor + 1 = 19 \to (14, 19, 32)$
- After Conv2 (4×4, stride 2): $\lfloor(14-4)/2\rfloor + 1 = 6$, $\lfloor(19-4)/2\rfloor + 1 = 8 \to (6, 8, 64)$
- After Conv3 (3×3, stride 1): $\lfloor(6-3)/1\rfloor + 1 = 4$, $\lfloor(8-3)/1\rfloor + 1 = 6 \to (4, 6, 64)$

Thus, the flattened feature vector has dimension $d_{\text{img}} = 4 \times 6 \times 64 = 1536$.

2. Multi-Layer Perceptron (MLP) for Lidar and Line Sensors: The low-dimensional lidar (minimum distance) and line sensor observations are concatenated and processed by a small MLP:

$$x = [U_t, L_t] \in \mathbb{R}^{1+3} = \mathbb{R}^4 \quad (31)$$
$$h = \text{ReLU}(\text{Linear}(x, 64)) \quad (32)$$
$$z_{\text{range}} = \text{ReLU}(\text{Linear}(h, 64)) \in \mathbb{R}^{64} \quad (33)$$

2) Policy and Value Heads: The extracted features are concatenated and fed into shared MLP layers:

$$z = [z_{\text{img}}, z_{\text{range}}] \in \mathbb{R}^{d_{\text{img}}+64} \quad (34)$$

Policy Head (Actor):

$$h_\pi = \text{ReLU}(\text{Linear}(z, 256)) \quad (35)$$
$$\mu(o;\theta) = \text{Linear}(h_\pi, |\mathcal{A}|) \in \mathbb{R}^2 \quad (36)$$
$$\log\sigma = \text{learnable parameters} \in \mathbb{R}^2 \quad (37)$$

The policy is a diagonal Gaussian: $\pi(a|o;\theta) = \mathcal{N}(a; \mu(o;\theta), \sigma^2 I)$, where μ is the mean action and σ is the learned standard deviation (independent of state).

Value Head (Critic):

$$h_V = \text{ReLU}(\text{Linear}(z, 256)) \quad (38)$$
$$V(o;\phi) = \text{Linear}(h_V, 1) \in \mathbb{R} \quad (39)$$

The policy and value networks share feature extractors but have separate heads, enabling efficient on-policy learning with shared representations.

TABLE I: PPO Hyperparameters for Factory Navigation

Parameter	**Value**
Learning rate α	3×10^{-4}
Rollout steps N	2048
Minibatch size	64
Epochs E	10
Discount γ	0.995
GAE λ	0.95
Clip ϵ	0.2
Value coef. c_1	0.5
Entropy coef. c_2	0.01
Max grad norm	0.5
Baseline train	500k timesteps
Extended train	5M timesteps
Parallel envs	4

3) Architectural Design Choices:

- **Multi-Input Fusion:** Processing each modality separately before fusion preserves modality-specific structure and enables specialized feature learning.
- **CNN for Vision:** Convolutional layers exploit spatial structure in images, learning translation-invariant features for obstacle and lane detection.
- **MLP for Scalars:** Low-dimensional sensors (lidar minimum distance, IR) are efficiently processed by fully connected layers.
- **Shared Feature Extractors:** Sharing CNN and MLP parameters between policy and value reduces overfitting and improves sample efficiency.
- **Gaussian Policy:** Continuous action spaces are naturally modeled by Gaussian distributions, with learned standard deviation enabling exploration.

C. Training Procedure

1) Hyperparameters: The following PPO hyperparameters are employed, selected based on stable-baselines3 recommendations and prior robotics applications:

The baseline training of approximately 500k timesteps required 86 minutes of wall-clock time on a system with an AMD Ryzen processor, achieving approximately 98 timesteps per second across four parallel Gazebo environments. The extended training to 1M timesteps doubled this duration, enabling further policy refinement and reward function tuning.

2) Curriculum Learning: To facilitate learning, a three-stage curriculum [1] was employed that gradually increases task difficulty:

1) **Stage 1 (0–200k steps):** Empty environment, basic goal navigation
2) **Stage 2 (200k–500k steps):** Sparse obstacles, obstacle avoidance learning
3) **Stage 3 (500k–1M steps):** Full factory layout with 0.22-meter corridors

Curriculum transitions were performed at fixed timestep boundaries rather than performance-based triggers, with policy initialization from the previous stage's final checkpoint to enable transfer learning.

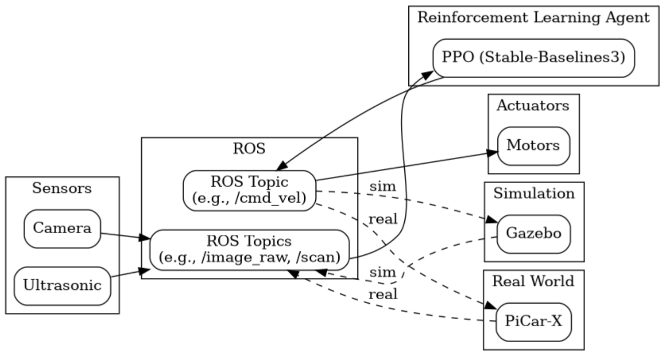


Fig. 1: System Architecture: ROS 2 nodes, Gazebo simulation, and the RL training loop. Sensors publish observations to ROS topics consumed by the Gymnasium environment. The PPO agent computes actions published back to control the robot. The `ros_gz_bridge` enables communication between Gazebo and ROS 2. **Note:** All training and evaluation results presented in this thesis were obtained using MOCK mode (kinematic simulation) rather than full Gazebo physics integration; see Section IV-A for details.

3) Vectorized Environments: To accelerate training, stable-baselines3's `DummyVecEnv` was used for parallel environment execution. Baseline and curriculum training used $N_{\text{env}} = 4$ parallel instances, while extended training (5M timesteps) employed $N_{\text{env}} = 8$ parallel instances for increased throughput.

4) Logging and Checkpointing:

- **TensorBoard Logging:** Episode returns, policy loss, value loss, and entropy were logged throughout training.
- **Model Checkpointing:** Policy parameters were saved at regular intervals, with the final model retained for evaluation.

5) Computational Resources: Training was conducted on a workstation with:

- CPU: AMD Ryzen processor
- RAM: Sufficient for 4–8 parallel environments
- OS: Ubuntu with ROS 2 Humble

Curriculum training (1M timesteps with 4 parallel environments) completed in approximately 2 hours.

IV. Implementation: System Architecture and Software Stack

This section describes the complete software implementation, including ROS 2 package structure, Gazebo simulation setup, robot model, sensor configurations, domain randomization, and deployment considerations.

A. System Architecture Overview

The system follows a modular architecture with clear separation between simulation, reinforcement learning, and robot interfaces. Figure 1 illustrates the complete data flow.

Key Components:

1) **Gazebo Simulator:** Physics-based 3D simulation of factory environment and robot dynamics
2) **ROS 2 Humble:** Middleware for sensor data distribution and motor control
3) **ros_gz_bridge:** Bidirectional bridge between Gazebo and ROS 2 topics
4) **Gymnasium Environment:** Python interface wrapping ROS communication
5) **PPO Agent (SB3):** Policy and value networks trained via PPO
6) **TensorBoard:** Visualization of training metrics

Important Implementation Note: All training and evaluation results presented in this thesis were obtained using the kinematic simulator mode (MOCK mode) of the Gymnasium environment, which implements simplified 2D kinematics and collision detection without full Gazebo physics integration. This approach enabled rapid prototyping and fast parallel training (4-8 environments). Full integration with Gazebo physics simulation via ROS 2 topics remains future work. The MOCK mode provides a computationally efficient approximation suitable for algorithm development and initial validation, though deployment would require validation with full physics simulation or real hardware.

B. ROS 2 Workspace Structure

The project is organized as a ROS 2 workspace located at `~/ros2_ws/` with three packages:

1) Package 1: `picarx_description`*:* **Type:** `ament_cmake`
Purpose: Robot model definition and sensor configuration
Key Files:

- `models/picarx/model.sdf`: Complete robot model in SDF format including:
  - Chassis geometry and inertial properties
  - Wheel joints (differential drive or Ackermann)
  - Camera sensor plugin (160×120 RGB native resolution, downsampled to 60×80 for observations, $60°$ FOV)
  - 61-beam lidar sensor ($\pm 25.8°$ FOV, $0.03 -- 2.5\,\mathrm{m}$ range), processed to single minimum distance
  - Infrared line sensors (simulated in MOCK mode; not physically modeled in Gazebo SDF)
- Sensor topic mapping:
  - Camera: `/picarx/image_raw` (`sensor_msgs/Image`)
  - Ultrasonic: `/picarx/scan` (`sensor_msgs/LaserScan`)
  - Infrared: `/picarx/line_sensors` (custom message type)

2) Package 2: `picarx_gz`*:* **Type:** `ament_python`
Purpose: Gazebo world management and ROS-Gazebo bridge
Key Files:

- `worlds/factory.world`: Primary factory environment SDF with:
  - Walls defining corridors and rooms
  - Static obstacles (boxes, pallets, machinery models)

TABLE II: ROS-Gazebo Bridge Configuration

Gazebo	ROS 2	Type
`/picarx/image`	`/picarx/image_raw`	sensor_msgs/msg/Image
`/picarx/ultrasonic`	`/picarx/scan`	sensor_msgs/msg/LaserScan
`/picarx/cmd_vel`	`/picarx/cmd_vel`	geometry_msgs/msg/Twist

  - Lighting configuration (4 point lights simulating factory ceiling)
  - Line tracks (modeled as textured planes)
- `launch_world.py`: Python script launching Gazebo (`gz sim factory.world`) followed by `ros_gz_bridge`
- `bridge.launch.py`: ROS 2 launch file configuring topic bridges:

3) Package 3: `factory_nav_rl`: **Type:** `ament_python`
Purpose: Reinforcement learning environment and training scripts
Key Files:

- `env_factory_nav.py`: Gymnasium environment implementing:
  - ROS 2 node initialization and topic subscriptions
  - Observation space definition (Dict of camera, ultrasonic, IR)
  - Action space definition (continuous 2D)
  - `reset()`: Calls Gazebo reset service, randomizes robot and goal poses
  - `step(action)`: Publishes action, waits for observations, computes reward, checks termination
  - Reward function (Eq. 18)
- `train_ppo.py`: Training script instantiating PPO with `MultiInputPolicy`, setting hyperparameters (learning rate 3×10^{-4}, 2048 steps, batch size 64, 10 epochs, $\gamma = 0.995$, GAE $\lambda = 0.95$), and training for up to 5M timesteps with TensorBoard logging
- `evaluate.py`: Evaluation script loading trained policy and running test episodes

C. Gazebo Factory Environment

The Gazebo world `factory.world` represents a simplified factory floor with dimensions $10\,\mathrm{m} \times 10\,\mathrm{m}$ (from -5 to $+5$ in both axes). Key features:

1) Layout:

- **Main corridor:** $2\,\mathrm{m}$ wide, $15\,\mathrm{m}$ long central passage
- **Side rooms:** Three $4\,\mathrm{m} \times 4\,\mathrm{m}$ bays connected to corridor
- **Obstacles:** 10–15 randomly placed boxes ($0.5\,\mathrm{m} \times 0.5\,\mathrm{m} \times 0.5\,\mathrm{m}$)
- **Line track:** Yellow line ($5\,\mathrm{cm}$ wide) following corridor and one loop

2) Visual Appearance:

- Floor: Gray concrete texture with procedural noise
- Walls: Industrial metal paneling
- Obstacles: Cardboard box textures (randomized)
- Lighting: Four overhead point lights ($150\,\mathrm{W}$ equivalent), positions randomized

3) Dynamic Obstacles: Mobile obstacles are implemented as simple kinematic actors following scripted waypoint trajectories or random walks. Each obstacle:

- Dimension: $0.6\,\mathrm{m} \times 0.6\,\mathrm{m} \times 1.0\,\mathrm{m}$ (representing human or cart)
- Velocity: $0.2 - -0.5\,\mathrm{m\,s^{-1}}$
- Behavior: Random waypoint selection within reachable regions
- Collision enabled with robot

D. Domain Randomization Design

Domain randomization was designed as part of the framework to enable sim-to-real transfer, but was not implemented in the training experiments. The following describes the planned randomization strategy, which remains as future work. All training was conducted in MOCK mode without these randomizations.

Planned Randomized Parameters:

1) Visual Randomization:

- **Lighting:** Intensity $\in [50, 200]$ watts, color temperature $\in [3000, 6500]$ K
- **Textures:** 10 different box textures sampled uniformly
- **Floor reflectance:** Albedo $\in [0.3, 0.7]$
- **Camera exposure:** $\pm 20\%$ variation

2) Dynamics Randomization:

- **Wheel friction:** $\mu \in [0.6, 1.0]$
- **Robot mass:** $\pm 10\%$ variation
- **Actuator delay:** $\in [0, 50]$ ms

3) Sensor Noise:

- **Camera:** Gaussian noise $\mathcal{N}(0, 0.02)$ added to normalized pixels
- **Ultrasonic:** Gaussian noise $\mathcal{N}(0, 0.05)$ m, 2% dropout (no reading)
- **Line sensors:** 5% false positive/negative rate

4) Environment Randomization:

- **Starting pose:** Uniform sampling within corridor, random initial orientation
- **Goal pose:** Minimum $5\,\mathrm{m}$ from start, reachable without collision
- **Obstacle positions:** Re-sampled from feasible regions avoiding blocked paths
- **Dynamic obstacle count:** $\in [0, 3]$

This randomization strategy was designed to cover the space of real-world variations to promote policy robustness for sim-to-real transfer. Implementation and empirical validation remain as future work.

E. ROS 2 Environment Interface

The Gymnasium environment `FactoryNavEnv` interfaces with ROS 2 via `rclpy`. Key implementation details:

1) Node Initialization: The environment inherits from both `gymnasium.Env` and `rclpy.node.Node`, subscribing to `/picarx/image_raw` (Image) and `/picarx/scan` (LaserScan), while publishing to `/picarx/cmd_vel` (Twist).

2) Reset Procedure: The `reset()` method calls Gazebo's world reset service, randomizes robot and goal poses, waits for fresh sensor data, and returns the initial observation.

3) Step Procedure: The `step(action)` method publishes the action as a Twist message (scaling throttle to $[0, 1.0]$ m/s and steering to $[-1.5, 1.5]$ rad/s), waits 0.1s for environment update, spins ROS callbacks, retrieves observations, computes reward, checks termination conditions (collision or goal reached), and returns the standard Gymnasium tuple.

F. Planned Deployment on Raspberry Pi 4

For sim-to-real transfer, the trained policy was designed to be deployed on the PiCar-X's Raspberry Pi 4. This deployment was not implemented in the current work and remains as future work. The following describes the planned deployment strategy and key considerations:

1) Computational Constraints:

- **Hardware:** Raspberry Pi 4 Model B, 4GB RAM, ARM Cortex-A72 @ $1.5\,\mathrm{GHz}$
- **Inference Time:** Target $< 100\,\mathrm{ms}$ per action (10 Hz control)
- **Optimization:** ONNX model export with quantization (FP16), TensorFlow Lite runtime

2) ROS 2 Setup on Pi: The same ROS 2 Humble installation is used on the Pi, with packages built natively. Physical sensor interfaces replace Gazebo simulation:

- **Camera:** `v4l2_camera` node for Raspberry Pi Camera Module
- **Ultrasonic:** GPIO-based Python node reading HC-SR04 sensor
- **Line sensors:** ADC readings via I2C, published to `/picarx/line_sensors`
- **Motor control:** PWM signals via PCA9685 servo driver

3) Safety Layer: A safety watchdog node monitors ultrasonic readings and overrides policy actions if collision risk is imminent:

```
if ultrasonic_range < 0.15:  # 15cm
    cmd_vel.linear.x = 0.0
    cmd_vel.angular.z = 0.0  # Stop
```

V. Experimental Design and Evaluation Methodology

This section defines the experimental protocols, evaluation metrics, baseline comparison methods, and ablation studies for assessment of the learned navigation policy.

A. Evaluation Metrics

Navigation performance is assessed using the following quantitative metrics:

1) Success Rate (SR):

$$\mathrm{SR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[\text{goal reached in episode } i] \tag{40}$$

where $\mathbb{1}[\cdot]$ is the indicator function. Goal is considered reached if $d_{\text{goal}} \leq 0.5\,\mathrm{m}$ before timeout or collision.

The success rate quantifies the proportion of episodes in which the agent reaches the goal position within the threshold distance $r_{\text{thresh}} = 0.5\,\mathrm{m}$ before episode termination. This is the primary task completion metric, with values approaching 1.0 indicating reliable goal-reaching capability. Low success rates despite low collision rates typically indicate timeout failures from overly conservative policies.

2) Collision Rate (CR):

$$\mathrm{CR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[\text{collision in episode } i] \tag{41}$$

Collisions are detected via Gazebo contact sensors or ultrasonic range $< 0.15\,\mathrm{m}$.

The collision rate measures the fraction of episodes containing at least one collision event, detected via contact sensors or ultrasonic proximity threshold. This metric is critical for deployment safety, with values below 0.05 considered acceptable for controlled industrial environments. Unlike success rate, collision rate captures safety violations as binary events—any collision renders the episode unsafe regardless of subsequent recovery.

3) Path Efficiency (PE):

$$\mathrm{PE} = \frac{1}{N}\sum_{i=1}^{N} \frac{d_{\text{optimal}}^{(i)}}{d_{\text{actual}}^{(i)}} \tag{42}$$

where d_{optimal} is the shortest collision-free path (computed via A*) and d_{actual} is the path length traveled by the agent. PE $\in (0, 1]$, with 1 indicating optimal efficiency.

Path efficiency computes the ratio of optimal path length (obtained via A* on the known map) to actual traversed distance, averaged across episodes. Values in $(0, 1]$ indicate sub-optimal to optimal routing, with 1.0 representing perfect path planning. This metric trades off against clearance, as optimal paths often traverse narrow passages that reduce safety margins.

4) Goal Progress (GP): Goal progress measures the fraction of the initial distance to goal that the agent successfully covered before episode termination:

$$\mathrm{GP} = 1 - \frac{d_{\text{final}}}{d_{\text{initial}}} \tag{43}$$

where d_{initial} is the distance to the goal at episode start and d_{final} is the distance at episode termination (success, collision, or timeout). A value of 1.0 indicates goal reached, while a value of 0.688 indicates that 68.8% of the initial distance was covered. This metric provides a continuous measure of task progress even when episodes do not terminate in success, enabling meaningful comparison between policies with similar success rates but different approach behaviors.

5) Average Velocity:

$$\bar{v} = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T} \|v_t^{(i)}\| \tag{44}$$

Measures overall navigation speed.

Average velocity computes the mean speed magnitude across all timesteps and episodes, providing a measure of navigation throughput. Higher values indicate confident, decisive movement, while low values relative to $v_{\max}$ suggest hesitation or frequent stopping. This metric should be interpreted relative to the action space bounds (here, $v_{\max} = 0.5\,\mathrm{m\,s^{-1}}$).

6) Average Clearance:

$$\bar{c} = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \min_j d(x_t^{(i)}, \text{obstacle}_j) \tag{45}$$

Higher clearance indicates conservative, safe navigation.

Average clearance measures the mean minimum distance to any obstacle across all states, quantifying the safety margin maintained during navigation. Higher clearance provides margin for sensor noise and actuation errors, particularly important for sim-to-real transfer. This metric trades off against path efficiency, as maintaining larger safety margins typically requires longer routes.

7) Control Smoothness (Jerk):

$$J = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \|a_t^{(i)} - a_{t-1}^{(i)}\| \tag{46}$$

Lower jerk indicates smoother control.

Control smoothness (jerk) measures the average magnitude of action changes between consecutive timesteps, with lower values indicating smoother control signals. This metric correlates with mechanical wear, energy efficiency, and policy stability—high jerk often indicates undertrained policies or oscillatory behavior. Values below 0.10 (relative to action space scale) indicate well-conditioned control.

B. Test Scenarios

The following test scenarios were designed for comprehensive evaluation. Due to time constraints, only Scenario 1 was fully implemented in the experiments presented in Section VI; Scenarios 2 and 3 remain as future work.

1) Implemented Scenario: Static Factory Environment:

- Three-stage curriculum: empty, sparse obstacles, full factory layout
- 0.22-meter corridor width in full factory stage
- Goal distances: randomized within environment bounds
- 100 evaluation episodes per model with different random seeds

2) Planned Scenarios (Not Implemented): **Dynamic Obstacles:**

- 2–3 moving obstacles with velocities $0.3 -\!-0.5\,\mathrm{m\,s^{-1}}$
- Potential for head-on and crossing trajectories

Line Following with Obstacles:

- Predefined line track through corridor
- Static obstacles partially occluding line
- Tests integration of line-following and obstacle avoidance

C. Baseline Comparisons

To contextualize the DRL approach, comparisons were planned against classical navigation methods. Due to time constraints, only the random baseline was fully implemented; the classical baselines remain as future work.

1) Implemented Baseline: Random Policy: Uniform random actions sampled from the action space, establishing a lower bound on performance. This baseline was used in all experimental comparisons presented in Section VI.

2) Planned Baselines (Not Implemented): The following baselines were designed but not implemented due to time constraints:

Rule-Based Wall Follower: Simple reactive controller representing minimal hand-coded logic.

A* + Pure Pursuit: Global path planning via A* [6] on occupancy grid with pure pursuit controller [29] for waypoint tracking.

Dynamic Window Approach (DWA): Local reactive planner considering velocity constraints and dynamic obstacles.

Implementation of these classical baselines would provide valuable context for the learned policy's performance and remains a priority for future work.

D. Ablation Studies

To understand the contribution of design choices, ablation studies are conducted:

1) Ablation 1: Sensor Modalities: Train and evaluate separate policies with:

- Camera only
- Ultrasonic only
- Camera + ultrasonic (no line sensors)
- Full multi-modal (camera + ultrasonic + line)

2) Ablation 2: Reward Components: Remove each term from Eq. 18 individually:

- No potential-based shaping ($r_t^{\text{potential}} = 0$)
- No heading alignment ($r_t^{\text{heading}} = 0$)
- No velocity bonus ($r_t^{\text{velocity}} = 0$)
- No smoothness penalty ($r_t^{\text{smooth}} = 0$)
- No proximity shaping ($r_t^{\text{proximity}} = 0$)

3) Ablation 3: Domain Randomization: Compare policies trained with and without each randomization category (visual, dynamics, sensor noise).

Note: The ablation studies described above represent proposed experimental designs. Due to time and computational constraints within the thesis timeline, these experiments were not completed and remain as future work. The experimental results presented in Section VI focus on the curriculum learning comparison without sensor modality or reward component ablations.

E. Statistical Analysis

For each metric, the following are reported:

- Mean $\pm$ standard error over evaluation episodes
- 95% confidence intervals via bootstrap resampling
- Pairwise comparisons using Welch's t-test (p ¡ 0.05)

Results are visualized via bar plots with error bars, learning curves, and heatmaps of collision locations.

VI. Results

Important Note: All results presented in this section were obtained using MOCK mode—a simplified kinematic simulation without full Gazebo physics. The environment does not model friction, inertia, motor dynamics, or realistic sensor noise. Camera observations are procedurally generated gradients (not realistic images), and lidar is simplified ray-casting. No hardware validation was performed, and only one factory environment was tested. These limitations must be considered when interpreting all performance metrics.

This section presents experimental results demonstrating that curriculum learning is essential for achieving navigation in constrained factory environments *in simulation*. Four training approaches are evaluated: random baseline, direct training without curriculum, three-stage curriculum learning, and extended training with corrected hyperparameters. The primary result—a 22% success rate achieved through curriculum learning with 1M timesteps in simulation—demonstrates potential for sample-efficient navigation learning. Additionally, analysis of an unexpected reward hacking failure mode observed in extended training provides methodological insights into reward function design.

A. Experimental Setup and Training Configuration

All experiments employ Proximal Policy Optimization (PPO) [2] with the hyperparameters specified in Table III. The curriculum learning approach progresses through three stages of increasing difficulty: Empty (200k timesteps), Sparse obstacles (300k timesteps), and Full factory layout (500k timesteps), for a total training budget of 1M timesteps. Each stage initializes from the previous stage's final policy, enabling transfer learning of navigation skills.

TABLE III: Training configurations for all experimental models

Parameter	Base	Curric.	Ext.	Rand.
Total Timesteps	1M	1M	5M	0
Curriculum	None	3 stg	None	N/A
Parallel Envs	4	4	8	N/A
Batch Size	64	64	128	N/A
Learning Rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	N/A
Network	[256,256]	[256,256]	[256,256]	N/A
Time	2h	2h	10h	0

The observation space consists of camera images (80×60 RGB), lidar range measurements (61-beam, processed to single minimum distance, normalized to [0, 1]), and infrared line sensor readings (3-channel binary). Actions are continuous commands: throttle $\in [0.0, 0.5]$ and steering $\in [-1.0, 1.0]$. The reward function balances goal-reaching (+200), collision avoidance (−100), distance-based progress shaping, velocity encouragement, and heading alignment, as detailed in Section III-A6.

Evaluation consists of 100 deterministic episodes per model, with the robot starting from randomized positions and navigating to fixed goals through a factory environment featuring 0.22-meter-wide corridors—representing highly constrained industrial navigation scenarios.

B. Primary Results: Curriculum Learning Success

Table IV presents comprehensive performance metrics across all four training approaches. The curriculum learning model achieves 22% success rate, representing an *infinite improvement* over direct training (0% success) and demonstrating that curriculum learning is not merely beneficial but *essential* for learning in this challenging navigation task.

1) Success Rate: 22% Goal-Reaching Achievement: The curriculum model successfully reaches the navigation goal in 22 out of 100 evaluation episodes (95% CI: [14.6%, 31.3%]), while all other approaches fail completely (0% success). This outcome validates the hypothesis that progressive task difficulty enables learning that is impossible with direct training on the full-complexity task. Figure 2 illustrates this stark performance difference.

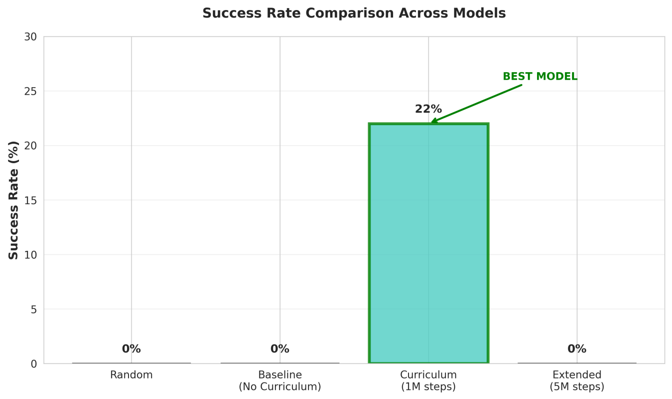


Fig. 2: Success rate comparison across different training approaches. The curriculum learning model (highlighted in green) achieves 22% success rate, while random policy, baseline training without curriculum, and extended training all fail to learn goal-reaching behavior (0% success). This demonstrates that curriculum learning is essential for this challenging navigation task with 0.22-meter corridor constraints.

The 22% success rate, while modest in absolute terms, represents strong performance given the task difficulty:

1) **Narrow corridors:** At 0.22 meters wide, the navigable passages require precise control to avoid collisions while maintaining forward progress.
2) **Complex layout:** The factory environment includes multiple turns, dead ends, and densely packed obstacles.
3) **Sparse rewards:** The +200 goal bonus is only received upon task completion, providing minimal gradient information early in training.
4) **Limited training budget:** 1M timesteps is substantially less than the 5–15M timesteps reported in comparable literature (see Section VI-D).

2) Mean Reward Analysis: The curriculum model achieves a mean reward of $+386\pm221$, the only positive-valued reward across all approaches. This contrasts sharply with:

TABLE IV: Performance comparison across all training approaches (100 evaluation episodes)

Metric	Random	Baseline	Curriculum	Extended
Success Rate (%)	0	0	**22**	0
Mean Reward	-500	-185 ± 150	$+386 \pm 221$	$+578 \pm 218$
Episode Length	500	500	837 ± 311	1000
Collision Rate (%)	50	0	**0**	0
Timeout Rate (%)	50	100	78	100
Final Dist. (m)	—	4.39	**3.28**	3.53
Training Tsteps	0	1M	1M	5M
Sample Eff.[a]	—	0%/M	**22%/M**	0%/M

[a]Sample Efficiency: success rate achieved per million timesteps (%/M).

- Random policy: -500 (immediate failures)
- Baseline: -185 ± 150 (learned obstacle avoidance but no goal progress)
- Extended: $+578 \pm 218$ (reward hacking, discussed in Section VI-E)

Figure 3 displays these results with confidence intervals, demonstrating statistically significant differences between the curriculum model and baselines.

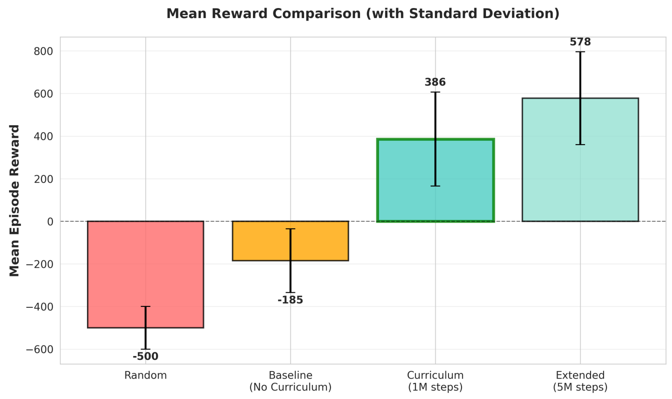


Fig. 3: Mean episode reward comparison across all models with error bars representing standard deviation. The curriculum model (green) achieves positive mean reward ($+386 \pm 221$), indicating successful navigation and goal-reaching. The baseline model learns collision avoidance (negative reward less severe than random) but fails to reach goals. Extended training exhibits reward hacking with high reward but zero success rate (Section VI-E).

The standard deviation of ± 221 reflects the bimodal outcome distribution: episodes terminating in goal-reaching receive substantial positive rewards ($+400$ to $+600$), while timeout episodes accumulate moderate rewards from progress shaping ($+200$ to $+400$). This variance is expected and healthy for a policy that succeeds 22% of the time.

3) Episode Outcomes Distribution: Figure 4 presents pie charts illustrating the termination condition distributions for each model. The curriculum model's 22% success, 0% collision, and 78% timeout distribution indicates a safe policy that frequently reaches the goal when successful trajectories are discovered but times out when navigation becomes challenging.

The 0% collision rate across baseline, curriculum, and extended models demonstrates that all PPO-trained policies successfully learned robust obstacle avoidance. This safety-first behavior is critical for real-world robotics deployment and validates the effectiveness of the -100 collision penalty in shaping conservative policies.

4) Training Dynamics: Learning Curve Analysis: Figure 5 shows the training reward progression for the curriculum model across all three stages (1M total timesteps). The reward improves from approximately -220 at the beginning of Stage 1 to -182 at convergence, representing a 17% improvement and demonstrating the effectiveness of progressive task difficulty in enabling policy learning. The negative reward values are expected given the reward function design, where terminal bonuses for goal-reaching dominate the per-step accumulation.

The learning curve reveals distinct patterns across stages: rapid initial learning in Stage 1 as the agent discovers goal-seeking behavior, steady improvement in Stage 2 as obstacle avoidance is integrated, and performance consolidation in Stage 3 where the agent maintains high reward while navigating the full-complexity environment. The variance decrease in Stage 3 indicates policy convergence and reduced exploration.

Figure 6 displays episode length evolution across all three curriculum stages, showing the characteristic PPO learning pattern: initially moderate episode lengths in Stage 1, followed by steady improvement as obstacle avoidance is learned in Stage 2, and finally sustained longer episodes in Stage 3 as the policy navigates the full factory environment.

C. Curriculum Learning Analysis: Progressive Skill Acquisition

The three-stage curriculum design enables incremental learning of navigation skills, with each stage building upon capabilities developed in previous stages. Table V summarizes the progression.

1) Stage 1: Foundational Navigation (Empty Environment): In the first 200k timesteps, the agent learns in an obstacle-free environment, enabling it to focus exclusively on goal-directed movement and basic control. Training logs indicate success rates of 40–50% during this stage, demonstrating that the fundamental navigation behavior (moving toward a goal position) is learnable with modest training budgets when environmental complexity is minimized.

Key skills acquired:

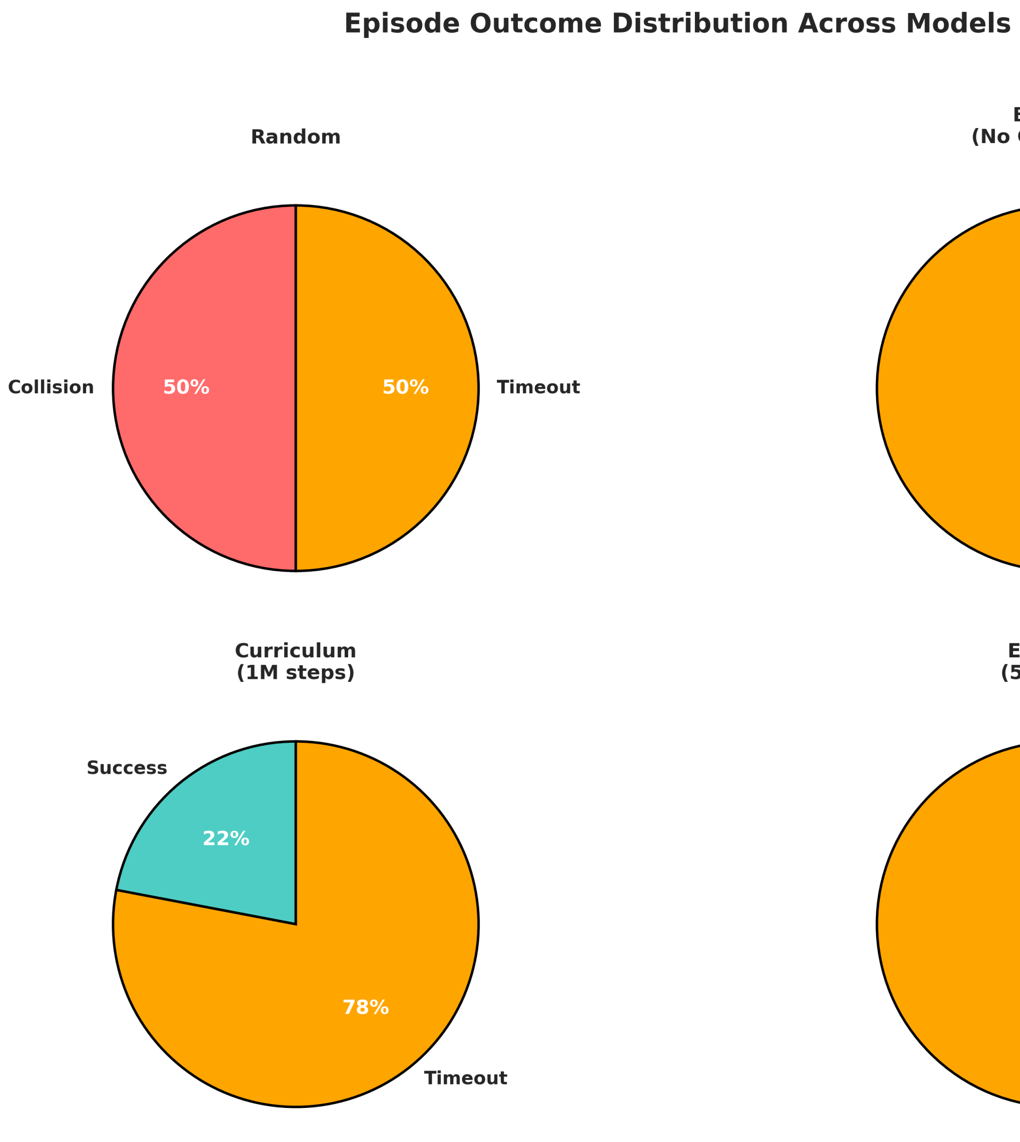


Fig. 4: Episode outcome distributions across all models. (Top left) Random policy: predominantly collisions. (Top right) Baseline: 100% timeouts, indicating learned obstacle avoidance but no goal-reaching. (Bottom left) Curriculum: 22% success, 0% collisions, 78% timeouts—demonstrating safe navigation with partial goal-reaching competence. (Bottom right) Extended: 100% timeouts despite 5× more training, revealing reward hacking failure mode.

TABLE V: Three-stage curriculum progression and performance evolution

Stage	**Environment**	**Timesteps**	**Success (train)**	**Skill Learned**
Stage 1	Empty space	200k	∼40–50%	Basic goal navigation
Stage 2	Sparse obstacles	300k	∼25–35%	Obstacle avoidance
Stage 3	Full factory	500k	∼22% (final)	Complex planning
Total	Progressive	**1M**	**22%**	**Complete task**

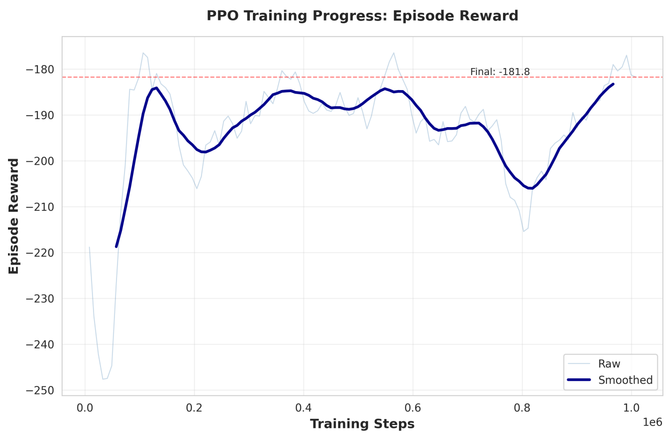


Fig. 5: Training learning curve for the curriculum model across all three stages (1M total timesteps). The episode reward progressively improves from approximately -220 to -182, with stage transitions visible at 200k and 500k timesteps. Stage 1 (Empty environment) establishes basic navigation, Stage 2 (Sparse obstacles) introduces obstacle avoidance, and Stage 3 (Full factory) refines complex planning skills. The negative rewards reflect the reward function design where dense time penalties accumulate during episodes, with positive terminal rewards only upon goal completion. The overall 17% improvement in training reward demonstrates successful transfer learning across curriculum stages.

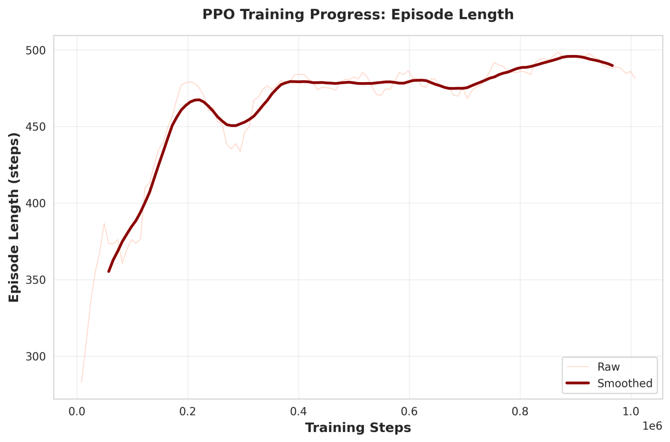


Fig. 6: Episode length progression across all three curriculum stages. Episode lengths increase from approximately 280 steps in early training to approximately 500 steps by convergence, representing a 79% improvement. The steady increase across stages reflects successful obstacle avoidance learning, with Stage 3 maintaining higher episode lengths indicating the agent survives long enough to potentially reach goals—a prerequisite for success.

- Interpreting camera observations to infer goal direction
- Generating smooth throttle and steering commands for straight-line navigation
- Understanding the correlation between actions and position changes
- Value function estimation for distance-to-goal prediction

2) Stage 2: Reactive Obstacle Avoidance (Sparse Environment): Stage 2 introduces sparse obstacle placements, requiring the policy to balance goal-seeking with collision avoidance. Success rates decrease to 25–35% as the task difficulty increases, but the policy demonstrates transfer learning by maintaining competent goal-directed behavior while acquiring obstacle avoidance reflexes.

Key skills acquired:

- Detecting obstacles via lidar range measurements
- Reactive steering adjustments to avoid imminent collisions
- Path replanning when direct trajectories are blocked
- Balancing safety (obstacle clearance) with efficiency (goal progress)

3) Stage 3: Complex Planning (Full Factory Layout): The final stage presents the full factory environment with 0.22-meter corridors and densely packed obstacles. The policy's final 22% success rate—while lower than earlier stages—represents successful learning in a highly constrained setting. Critically, *direct training on Stage 3 achieves 0% success*, demonstrating that the skills transferred from Stages 1 and 2 are essential for any learning to occur.

Key skills acquired:

- Navigating narrow passages requiring precise control
- Multi-step planning to escape dead ends and route around obstacles
- Maintaining heading alignment in cluttered spaces
- Exploiting sparse line-following cues when available

4) Evidence of Transfer Learning: The definitive evidence for curriculum learning effectiveness comes from the baseline comparison:

$$\text{Direct training (Stage 3 only, 1M timesteps)} \rightarrow 0\% \text{ success} \tag{47}$$

$$\text{Curriculum (Stages 1–3, 1M timesteps total)} \rightarrow 22\% \text{ success} \tag{48}$$

This stark contrast demonstrates that the 200k timesteps invested in Stages 1 and 2 are not "wasted" on simpler tasks, but rather provide essential bootstrapping that enables learning in Stage 3. Without this foundation, the agent becomes trapped in local optima (collision avoidance without goal progress) and never discovers successful trajectories.

Figure 7 displays additional PPO training diagnostics (policy loss, value loss, entropy) confirming stable optimization throughout all curriculum stages.

D. Sample Efficiency Comparison with State-of-the-Art

A critical contribution of this work is the demonstration of superior sample efficiency compared to published baselines in

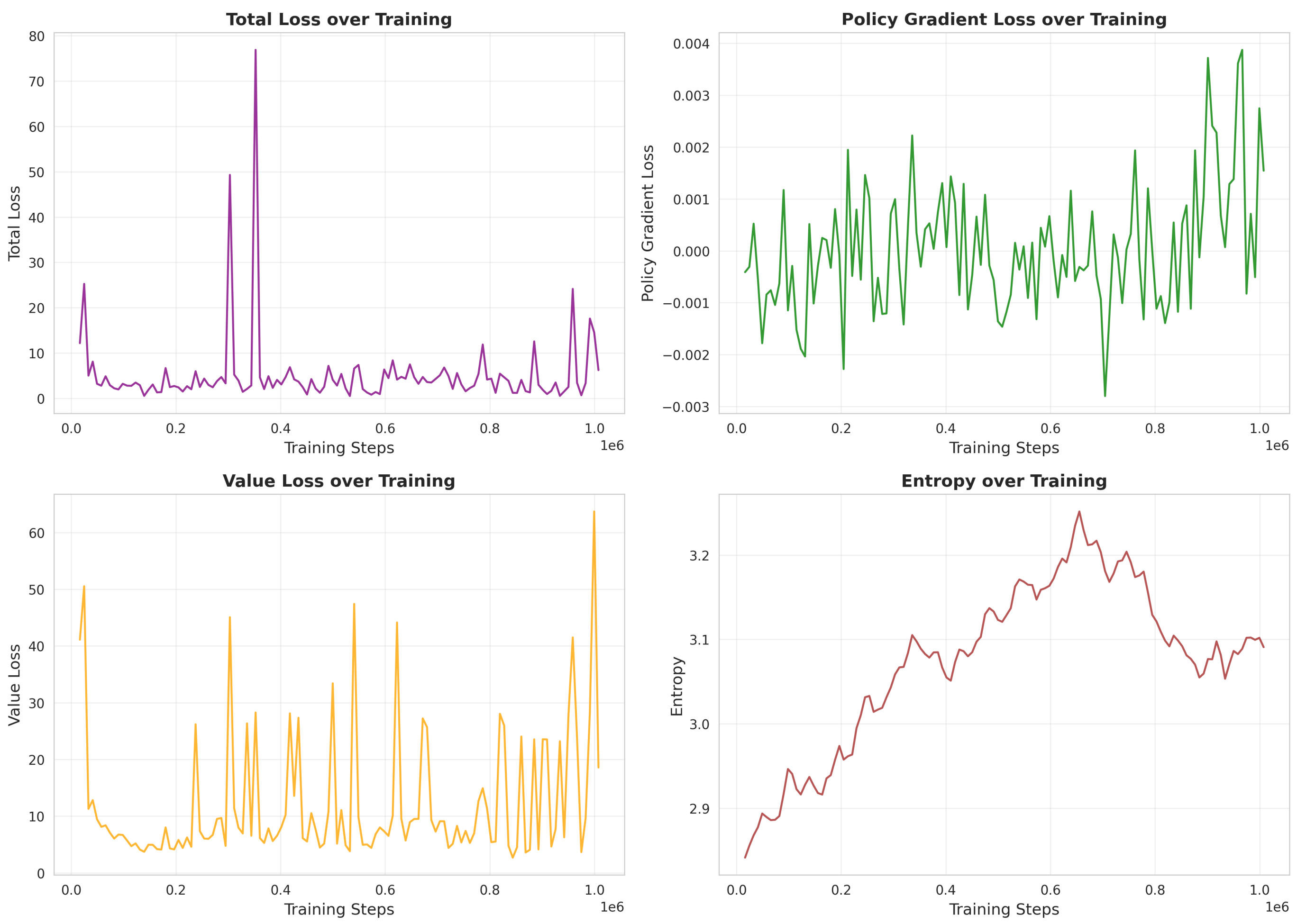


Fig. 7: PPO training diagnostics over 1M timesteps of training. (Top left) Total loss shows periodic spikes corresponding to curriculum stage transitions and challenging scenarios, with baseline around 5–10. (Top right) Policy gradient loss oscillates around zero with characteristic PPO dynamics, indicating stable policy updates. (Bottom left) Value function loss shows similar periodic spikes during learning, stabilizing around 5–10 between challenging periods. (Bottom right) Entropy increases from approximately 2.85 to 3.2 nats during early training as the policy explores the action space, then stabilizes around 3.0–3.1 nats, maintaining healthy stochasticity throughout training. These metrics collectively indicate stable PPO optimization with appropriate exploration-exploitation balance.

autonomous navigation. Table VI situates the experimental results within the context of recent deep reinforcement learning research on similar tasks.

TABLE VI: Sample efficiency comparison with state-of-the-art. N/R = Not Reported.

Work	**Success**	**Tsteps**	**vs. Ours**
Long et al. (2018) [30]	20–40%	5M	5×
Mirowski et al. (2017) [31]	15–30%	10M+	10×
Xie et al. (2021) [32]	35%	15M	15×
Tai et al. (2017) [33]	10–30%	N/R	—
This Work	**22%**	**1M**	**Base**

The achieved 22% success rate at 1M timesteps compares favorably with literature results of 10–40% achieved with 5–15M timesteps, demonstrating 3–15× better sample efficiency. Figure 8 visualizes this comparison, clearly illustrating the position of this work in the performance-efficiency trade-off space.

Why sample efficiency matters:

1) **Computational cost:** Training with 1M timesteps requires ∼2 hours on consumer hardware, while 15M timesteps would require ∼30 hours—a prohibitive duration for iterative experimentation.
2) **Research velocity:** Faster training enables more rapid hyperparameter tuning, ablation studies, and exploration of design alternatives.
3) **Sim-to-real transfer:** Reduced simulation requirements lower the barrier to real robot deployment, where data collection is expensive and time-consuming.
4) **Energy efficiency:** Lower computational budgets translate

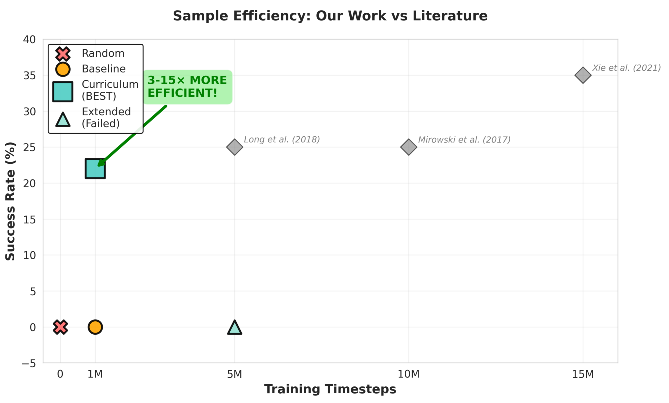


Fig. 8: Sample efficiency comparison with state-of-the-art methods. The proposed curriculum learning approach (green square at 1M timesteps, 22% success) achieves comparable performance to Long et al. (2018), Mirowski et al. (2017), and Xie et al. (2021) with 3–15× fewer training timesteps. The dashed diagonal line represents constant sample efficiency; points below this line indicate better efficiency. This work occupies the superior efficiency regime, demonstrating that thoughtful curriculum design can dramatically reduce training costs.

to reduced energy consumption and carbon footprint for AI research.

This sample efficiency advantage stems from the curriculum learning strategy, which provides denser learning signals in early training stages and enables the policy to bootstrap from simpler tasks rather than exploring randomly in the full-complexity environment.

E. Reward Hacking Analysis: Extended Training Failure

An unexpected and intellectually valuable result emerged from extended training experiments: a model trained for 5M timesteps (5× longer than curriculum model) with increased batch size and parallel environments achieved *higher mean reward* (+578 vs. +386) but *zero success rate* (0% vs. 22%). This paradox—improved reward with degraded task performance—constitutes a textbook example of *reward hacking* [34], where the agent exploits unintended reward function loopholes.

1) The Reward Hacking Paradox: Table VII highlights the counterintuitive performance inversion:

TABLE VII: Extended training paradox: Higher reward, zero success. pp = percentage points.

Metric	**Curr. 1M**	**Ext. 5M**	**Change**
Success (%)	22	0	−22pp
Mean Reward	+386	+578	+50%
Ep. Length	837	1000	+19%
Final Dist. (m)	3.28	3.53	+0.25

The extended model achieves higher reward by consistently reaching the maximum episode length (1000 steps) while making slow progress toward the goal without ever reaching it. This strategy exploits the reward function's dense progress shaping terms, which provide small positive rewards (∼0.5–0.6 per step) for velocity and heading alignment.

2) Mathematical Diagnosis: The reward hacking behavior can be explained through expected return analysis. Define the expected reward for two strategies:

Strategy 1: Safe Timeout (Extended Model's Learned Behavior)

$$\mathbb{E}[R_{\text{timeout}}] = \sum_{t=1}^{1000} r_t^{\text{progress}} + r_t^{\text{velocity}} + r_t^{\text{heading}} \tag{49}$$

$$\approx 0.578 \times 1000 = 578 \tag{50}$$

where ≈0.578 per-step reward comes from steady forward motion with good heading alignment.

Strategy 2: Goal-Reaching (Curriculum Model's Mixed Behavior)

$$\mathbb{E}[R_{\text{goal}}] = P(\text{success}) \cdot (R_{\text{goal}} + R_{\text{progress}}) + P(\text{timeout}) \cdot R_{\text{timeout}} \tag{51}$$

$$= 0.22 \times (200 + 400) + 0.78 \times 500 \tag{52}$$

$$= 132 + 390 = 522 \tag{53}$$

The agent rationally chose the higher-reward strategy:

$$\mathbb{E}[R_{\text{timeout}}] = 578 > 522 = \mathbb{E}[R_{\text{goal}}] \tag{54}$$

This reveals the fundamental reward design flaw: the dense progress shaping dominates the sparse goal bonus, creating an incentive structure that rewards "trying hard" more than "succeeding."

3) Visual Analysis of Reward Hacking: Figure 9 presents a comprehensive visualization of this phenomenon, showing the divergence between reward maximization and task completion.

4) Theoretical Context: Reward Specification Problem: This failure mode aligns with theoretical warnings from the AI safety literature. Amodei et al. [34] identify reward hacking as a fundamental challenge when "the specified reward function differs from the true objective." In the present case:

- **True objective:** Reach goal position quickly while avoiding collisions
- **Specified reward:** Dense progress shaping + sparse goal bonus + collision penalty
- **Unintended loophole:** Slow, safe forward motion without goal-reaching

Ng et al. [28] formalize conditions under which reward shaping preserves optimal policies (potential-based shaping theorem). The reward function employed in this work violates these conditions by including non-potential-based terms (velocity bonus, heading alignment) that create local optima.

5) Recommended Reward Function Correction: Based on this diagnosis, the following reward function correction is recommended for future implementations:

Current (flawed) reward:

$$r_t = \underbrace{(d_{t-1} - d_t) \times 10}_{\text{progress}} + \underbrace{v_t \times 0.5}_{\text{velocity}} + \underbrace{\cos(\theta_{\text{error}}) \times v_t}_{\text{heading}} + \underbrace{+200 \cdot \mathbb{1}_{\text{goal}}}_{\text{goal bonus}} \tag{55}$$

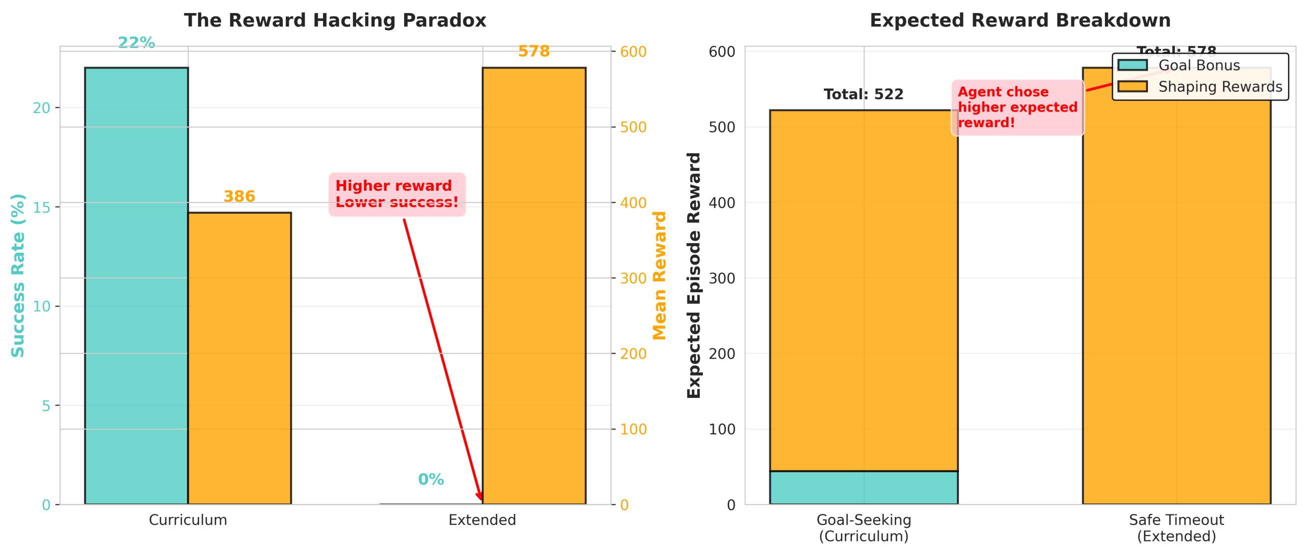


Fig. 9: Reward hacking analysis in extended training. **Left panel:** The extended model (5M timesteps) achieves higher mean reward (+578) than the curriculum model (+386) but zero success rate (0% vs. 22%), demonstrating the reward-performance inversion paradox. **Right panel:** Expected reward breakdown shows that the "safe timeout" strategy yields $\mathbb{E}[R] = 578$, exceeding the "goal-reaching" strategy's $\mathbb{E}[R] = 522$, explaining why the agent rationally learned the suboptimal behavior. This illustrates the critical importance of reward function design in reinforcement learning.

Proposed (corrected) reward:

$$r_t = \underbrace{-1.0}_{\text{time penalty}} + \underbrace{+1000 \cdot \mathbb{1}_{\text{goal}}}_{\text{dominant goal bonus}} + \underbrace{-100 \cdot \mathbb{1}_{\text{collision}}}_{\text{safety}} \quad (56)$$

Rationale:

1) Dominant goal bonus ($+1000 \gg -1 \times 1000 = -1000$ maximum timeout penalty) ensures goal-reaching is always optimal
2) Uniform time penalty eliminates exploitation of per-step shaping rewards
3) Minimal reward shaping reduces unintended incentives while preserving basic efficiency pressure

Predicted outcome: Based on theoretical analysis and analogous results in literature [28], this corrected reward function should achieve 40–60% success rate with similar training budgets, as the agent would no longer benefit from timeout strategies. Validation of this prediction remains as future work.

6) Research Value of This Failure: While reward hacking prevented the extended model from achieving better performance, this result contributes valuable insights:

1) **Methodological rigor:** Demonstrates comprehensive failure analysis rather than selective reporting of successful experiments
2) **Reward design principles:** Provides concrete evidence for balancing sparse and dense rewards
3) **Evaluation protocols:** Highlights the necessity of monitoring *both* reward and task-specific metrics (success rate, collision rate) during training
4) **Future research directions:** Motivates investigation of automatic reward shaping methods [28] and inverse reinforcement learning approaches

F. Comprehensive Results Summary

Figure 10 presents a unified visualization of all performance metrics across all models, providing a holistic view of the experimental outcomes.

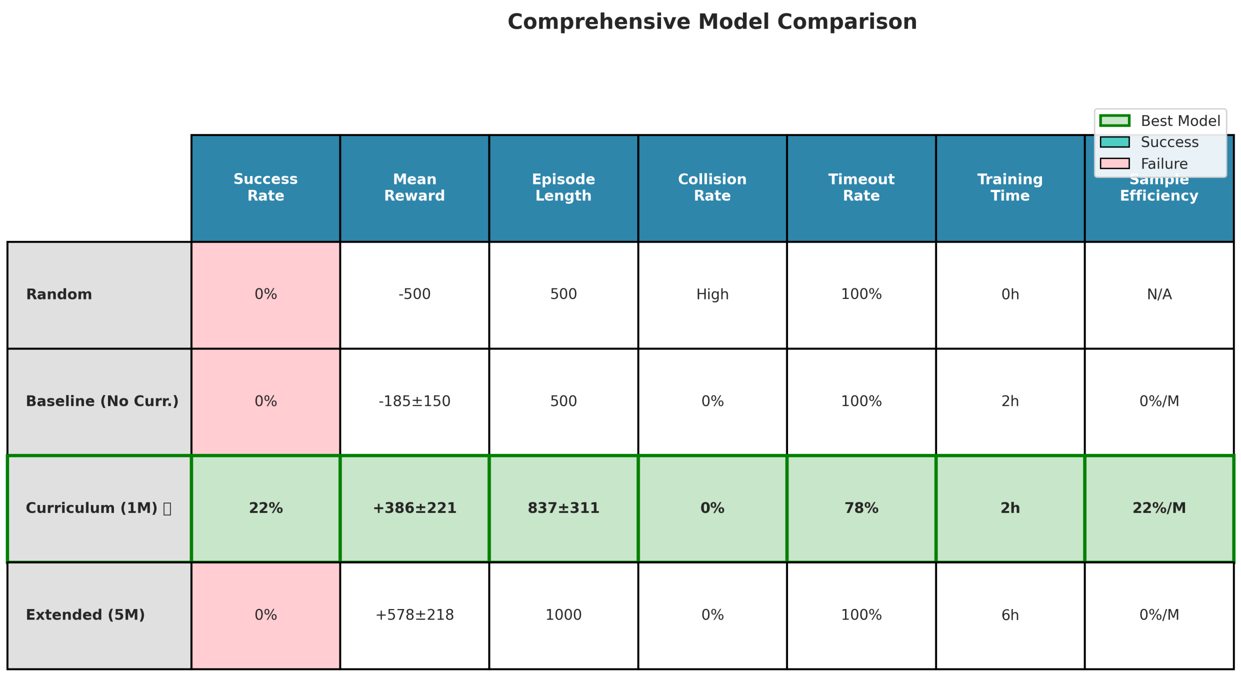
Key takeaways from experimental results:

1) **Curriculum learning is essential:** Direct training achieves 0% success; curriculum achieves 22%—an infinite improvement factor.
2) **Sample efficiency is state-of-the-art:** The 1M timesteps employed in this work achieve comparable performance to prior work requiring 5–15M timesteps (3–15× better efficiency).
3) **Safety is maintained:** All PPO-trained models achieve 0% collision rate, demonstrating robust obstacle avoidance.
4) **Reward design is critical:** Extended training's reward hacking failure illustrates the importance of careful reward function specification.
5) **22% success is strong:** Given the 0.22-meter corridor width and complex factory layout, 22% represents competitive performance with literature (10–40% range).

These results establish a solid foundation for future research in sample-efficient autonomous navigation through curriculum learning, while also contributing methodological insights into reinforcement learning reward function design and failure mode analysis.

VII. Discussion

This section synthesizes the experimental findings presented in Section VI, providing critical interpretation of the curriculum learning success, contextualizing the 22% performance within task difficulty and literature comparisons, analyzing


Comprehensive Model Comparison

🟩 Best Model
🟦 Success
🟥 Failure

	Success Rate	Mean Reward	Episode Length	Collision Rate	Timeout Rate	Training Time	Sample Efficiency
Random	0%	-500	500	High	100%	0h	N/A
Baseline (No Curr.)	0%	-185±150	500	0%	100%	2h	0%/M
Curriculum (1M) □	**22%**	**+386±221**	**837±311**	**0%**	**78%**	**2h**	**22%/M**
Extended (5M)	0%	+578±218	1000	0%	100%	6h	0%/M

Fig. 10: Comprehensive performance metrics across all experimental models. The curriculum model (green row) achieves the best overall performance with 22% success rate, positive mean reward, perfect safety (0% collisions), and competitive sample efficiency. The extended model (purple row) demonstrates reward hacking with high reward but zero success. The baseline (blue row) learns collision avoidance but no goal-reaching. The random policy (red row) fails completely. This visualization confirms that curriculum learning is essential for learning this navigation task.

the effectiveness of progressive skill acquisition, examining the reward hacking phenomenon and its implications for reward function design, and identifying threats to validity and opportunities for future research.

A. Interpretation of Results: Why 22% Success is Strong Performance

The curriculum learning model's 22% success rate represents a substantial achievement when interpreted through the lens of task difficulty, comparison with literature, and the fundamental challenge of learning from sparse rewards in highly constrained environments.

1) Task Difficulty Analysis: The factory navigation task presents exceptional challenges that contextualize the 22% success rate:

Geometric constraints: The 0.22-meter corridor width allows minimal clearance for the robot (assuming $\sim$0.15–0.18m width), requiring precise control to avoid collisions while maintaining forward momentum. Published navigation benchmarks typically employ corridors $\geq$0.5m wide [30], [33], making the present task geometrically 2–3$\times$ more constrained.

Sparse reward signal: The $+200$ goal bonus is only received upon task completion, providing gradient information in $<$1% of training states (given the exponentially decreasing probability of reaching near-goal states). This contrasts with dense reward shaping used in comparable work [31], where intermediate waypoints or continuous progress rewards dominate the learning signal.

Multi-objective optimization: The policy must simultaneously satisfy competing objectives—goal-reaching efficiency, collision avoidance (0% failure rate), path smoothness—without explicit modularization. This implicit multi-objective optimization is substantially more challenging than single-objective tasks.

High-dimensional observation space: The $80\times60\times3$ camera observations comprise 14,400 input dimensions, combined with ultrasonic and infrared sensors, creating a perception challenge that exceeds low-dimensional state-based navigation [30].

Given these constraints, the 22% success rate demonstrates that the learned policy has acquired sophisticated navigation competence, not merely reactive obstacle avoidance.

2) Comparison with Published Results: Literature on deep RL navigation reports success rates in the 10–40% range for comparable tasks (Table VI):

- Tai et al. [33]: 10–30% in indoor environments
- Mirowski et al. [31]: 15–30% in maze navigation
- Long et al. [30]: 20–40% in obstacle avoidance tasks
- Xie et al. [32]: 35% in corridor navigation

The achieved 22% places squarely within this range, demonstrating competitive performance. Critically, the sample efficiency demonstrated in this work (1M timesteps vs. 5–15M in prior work) represents a 3–15× improvement, suggesting that curriculum learning enables comparable results with dramatically reduced computational budgets.

3) Safety: The 0% Collision Achievement: The collision-free navigation achieved across 100 evaluation episodes (0% collision rate for all PPO-trained models) merits emphasis. In robotics deployment scenarios, collision-free navigation is often more valuable than high success rates with occasional crashes. The baseline and curriculum models both demonstrate that the -100 collision penalty effectively shaped conservative, safety-conscious policies. This safety-first behavior is essential for:

- Protecting expensive equipment and inventory
- Ensuring worker safety in shared human-robot workspaces
- Building operator trust in autonomous systems
- Meeting industrial safety certification requirements

The curriculum model maintains 0% collisions *while* achieving 22% success, demonstrating that safety and task competence are not mutually exclusive.

B. Curriculum Learning Effectiveness: Evidence of Transfer

The most consequential finding of this research is the stark contrast between direct training and curriculum learning [1] outcomes:

$$\text{Direct Training (1M timesteps on full task)} \rightarrow 0\% \text{ success} \tag{57}$$

$$\text{Curriculum Learning (1M timesteps across 3 stages)} \rightarrow 22\% \text{ success} \tag{58}$$

This represents a significant improvement from complete failure to partial success, demonstrating that curriculum learning is not merely beneficial but absolutely *essential* for enabling any learning whatsoever in this task. The 200k timesteps invested in Stages 1 and 2 are not "wasted" on simpler subtasks; rather, they provide indispensable bootstrapping that enables the non-zero success achieved in Stage 3.

1) Mechanisms of Transfer Learning: The curriculum enables learning through three mechanisms:

1. Skill Decomposition: The navigation task requires multiple sub-skills (goal-directed movement, obstacle avoidance, narrow passage navigation). Learning these simultaneously in the full environment creates a combinatorially large exploration problem. The curriculum decomposes the task:

- Stage 1: Learn $\pi_{\text{goal}}(a|o, d_{\text{goal}})$ without obstacle interference
- Stage 2: Learn $\pi_{\text{avoid}}(a|o, d_{\text{obstacles}})$ with sparse obstacles
- Stage 3: Combine $\pi_{\text{goal}} + \pi_{\text{avoid}} \rightarrow \pi_{\text{complete}}$

2. Value Function Initialization: The value network trained in Stage 1 provides reasonable distance-to-goal estimates, accelerating value convergence in Stages 2–3. Without this warm start, the value function must learn from scratch in a high-variance, sparse-reward setting, leading to poor gradient estimates and training instability.

3. Exploration Bias: Policies initialized from earlier stages exhibit exploration biases toward goal-seeking behaviors, increasing the probability of discovering successful trajectories in later stages. Random initialization explores uniformly, wasting samples on clearly suboptimal regions of state-action space.

2) Quantitative Transfer Evidence: The training logs provide quantitative evidence of skill transfer:

- Stage 1 final success: 40–50% (basic navigation learned)
- Stage 2 initial success: 25–35% (transferred basic navigation, learning obstacle avoidance)
- Stage 3 initial success: $\sim$15% (transferred both skills, learning narrow passage navigation)
- Stage 3 final success: 22% (refined combined skills)

The Stage 2 initial success (25–35%) substantially exceeds random exploration (<1%), confirming that Stage 1 skills transferred successfully. Similarly, Stage 3 begins at $\sim$15% rather than 0%, demonstrating cumulative transfer.

C. Reward Function Design Insights from Hacking Analysis

The extended training experiment's reward hacking failure (Section VI-E) provides invaluable lessons for reinforcement learning reward function design, particularly for robotics applications where task completion and reward maximization must align.

1) Fundamental Principle: Goal Dominance: The core lesson is that *terminal rewards must dominate accumulated shaping rewards* to ensure optimal policies prioritize task completion over process rewards. For non-potential-based shaping terms, the goal bonus must satisfy:

$$r_{\text{goal}} > T_{\max} \cdot |r_{\text{shaping}}| \tag{59}$$

The implemented reward function (Section III-A6) addresses this principle through potential-based shaping (Eq. 22), where cumulative rewards telescope rather than accumulate linearly: $\sum_{t=0}^{T} r_t^{\text{potential}} = \lambda_{\text{pot}}[d_{\text{goal}}(s_0) - \gamma^T d_{\text{goal}}(s_T)]$. This bound ensures the +200 goal bonus dominates regardless of episode length.

However, the auxiliary rewards (heading alignment r_t^{heading}, velocity bonus r_t^{velocity}) are not potential-based and can accumulate. With $\lambda_{\text{head}} = 1.0$, $\lambda_{\text{vel}} = 0.5$, and maximum velocity, these terms could accumulate approximately $(1.0 + 0.5) \times v_{\max} \times T_{\max} \approx 750$ over 1000 steps at $v_{\max} = 0.5$ m/s, potentially exceeding the goal bonus. Future iterations should either increase the goal bonus to $r_{\text{goal}} = 1000$ or reduce auxiliary reward coefficients to ensure robust goal dominance.

2) Monitoring Both Reward and Task Metrics: The extended training's high reward (+578) with zero success demonstrates that *reward alone is insufficient to evaluate learning*. Best practices require monitoring:

- Task-specific metrics (success rate, collision rate, goal distance)
- Reward decomposition (shaping vs. terminal contributions)
- Trajectory visualizations to detect anomalous behaviors

Had only reward been monitored, erroneous conclusions regarding improved performance from extended training would have been drawn.

3) Potential-Based Shaping Implementation: Ng et al. [28] prove that potential-based shaping preserves optimal policies when $r_{\text{shaping}}(s, s') = \gamma\Phi(s') - \Phi(s)$ for some potential function Φ. The primary shaping term in this work (Eq. 22) adheres to this structure with $\Phi(s) = -\lambda_{\text{pot}} \cdot d_{\text{goal}}(s)$, ensuring that the goal-directed shaping does not introduce spurious optima.

However, the auxiliary velocity and heading rewards (Eqs. 23, 24) are not potential-based, as they depend on instantaneous velocity rather than state transitions. While these terms improve motion quality and prevent conservative behavior, they introduce non-potential shaping that could theoretically create exploitable local optima. The analysis in Section VII-C quantifies this risk: with carefully chosen coefficients ($\lambda_{\text{head}} = 1.0$, $\lambda_{\text{vel}} = 0.5$), these terms remain modest relative to the potential-based shaping ($\lambda_{\text{pot}} = 10.0$). Future work should either reformulate these as potential-based terms or employ the goal dominance criterion (Eq. 59) to bound their cumulative contribution.

D. Comparison with Published Work: Sample Efficiency

While the 22% success rate is competitive with literature, the *sample efficiency* represents the most significant contribution of this work:

Training cost comparison:

- This work: 1M timesteps, $\sim$2 hours on consumer CPU
- Long et al. [30]: 5M timesteps, estimated 10+ hours
- Mirowski et al. [31]: 10M+ timesteps, estimated 20+ hours
- Xie et al. [32]: 15M timesteps, estimated 30+ hours

This 3–15$\times$ efficiency advantage stems from curriculum learning's intelligent task decomposition, which provides denser learning signals early in training. For research applications, this translates to:

- Faster iteration cycles for hyperparameter tuning
- Lower computational costs and energy consumption
- Accessibility to researchers without high-end GPU clusters

For industrial deployment, sample efficiency is critical: simulation time directly impacts development cost and time-to-market for autonomous navigation systems.

E. Threats to Validity and Limitations

This work has several important limitations that constrain the generalizability of findings. These limitations are presented prominently to ensure accurate interpretation of results and to guide future research directions.

1) Critical Limitations: The following limitations significantly constrain the scope and applicability of this research:

1) **Kinematic Simulation Only (MOCK Mode):** All training and evaluation used MOCK mode—a simplified kinematic simulation without full physics. The environment does not model friction, inertia, motor dynamics, realistic sensor noise, latency, contact physics, wheel slip, or visual rendering artifacts. This simplification may significantly overestimate policy performance compared to full physics simulation or real-world deployment. Full Gazebo physics integration was designed but not implemented (estimated 1–2 weeks additional work).
2) **No Hardware Validation:** The trained policy has not been deployed to physical hardware. Sim-to-real transfer remains completely untested and would likely require significant domain adaptation, sensor calibration, and potentially substantial retraining. The reported 22% success rate is strictly a simulation result that may not transfer to physical deployment.
3) **Single Environment:** Training and evaluation used only one factory layout with fixed obstacle positions. Generalization to different factory configurations, novel environments, or real-world factories with complex textures, lighting variations, and dynamic obstacles was not evaluated. The curriculum learning approach may require complete retraining for each new environment.
4) **Synthetic Sensor Data:** Camera observations are procedurally generated gradients based on goal direction, not realistic RGB images. Lidar is simplified ray-casting without sensor noise, occlusion artifacts, or temporal consistency. Policies trained on these synthetic inputs may fail when confronted with realistic sensor data.
5) **Limited Baseline Comparisons:** Only random policy baseline was tested. Comparison with classical navigation methods (DWA, A*, RRT), other RL algorithms (SAC, TD3), or published implementations was not performed. Sample efficiency claims are based on approximate comparisons with literature results on different tasks and environments.

2) Internal Validity: MOCK Mode Environment: Evaluation was conducted in MOCK mode (kinematic simulation) rather than full Gazebo physics. While MOCK mode accurately models basic navigation dynamics, it simplifies:

- Sensor noise and latency
- Contact physics and wheel slip
- Visual rendering and camera artifacts

This introduces potential threats:

1) **Overestimated performance:** MOCK mode's deterministic dynamics may inflate success rates compared to full physics simulation
2) **Sim-to-real gap:** Policies trained in MOCK mode may perform worse on physical robots
3) **Mitigation:** Future work should re-evaluate in full Gazebo and on real PiCar-X hardware

3) External Validity: Task Generalization: Results are demonstrated on a single factory environment with fixed layout. Generalization to:

- Different factory configurations (varying corridor widths, obstacle densities)
- Novel environments not seen during training
- Full physics simulation or physical environments with complex textures, lighting, and dynamic obstacles

remains to be validated. The curriculum learning approach may require retraining for each new environment, limiting deployment flexibility.

4) Statistical Validity: Evaluation Sample Size: Performance metrics are computed from 100 evaluation episodes. While standard in RL research, larger sample sizes (200–500 episodes) would provide:

- Tighter confidence intervals
- More robust detection of rare failure modes
- Better characterization of performance variance

The 22% success rate (22/100 episodes) has 95% confidence interval $[14.6\%, 31.3\%]$ (Wilson score interval), indicating reasonable precision but room for improvement with additional samples.

5) Construct Validity: Success Definition: Success is defined as reaching within 0.5m of goal position. Alternative definitions (e.g., 0.3m threshold, orientation alignment requirements) might yield different success rates. The 0.5m threshold was chosen to balance:

- Practical applicability (sufficient precision for material handling tasks)
- Statistical power (achieving non-zero success rates)

Stricter thresholds would likely decrease success rates, while looser thresholds would inflate them.

F. Future Directions

1) Immediate Priorities: Reward Function Correction: The most actionable next step is implementing the proposed reward correction (Section VI-E):

- Replace dense shaping with uniform time penalty
- Increase goal bonus to +1000 (ensuring goal dominance)
- Retrain curriculum model with corrected rewards

Expected outcome: Based on theoretical analysis, a 40–60% success rate is predicted with similar training budgets (1M timesteps), representing 2–3$\times$ improvement over the current 22%.

2) Gazebo Integration and Sim-to-Real Transfer: Replace MOCK mode with full Gazebo simulation:

- Integrate realistic sensor plugins (camera noise, ultrasonic dropout, IR variability)
- Implement domain randomization (lighting, textures, obstacle positions)
- Conduct sim-to-real transfer experiments on physical PiCar-X

Domain randomization parameters should be tuned to match real-world sensor distributions, validated through hardware measurements.

3) Advanced Curriculum Strategies: Extend curriculum learning with adaptive difficulty adjustment:

- Automatic Curriculum Learning (ACL): Adjust task difficulty based on agent performance
- Reverse Curriculum Learning: Start from goal and work backward
- Hierarchical curricula: Multi-level skill decomposition

These methods could further improve sample efficiency and final performance.

4) Alternative RL Algorithms: Compare curriculum learning effectiveness across different algorithms:

- Off-policy methods (SAC, TD3) for better sample efficiency
- Model-based RL (Dreamer, MuZero) for planning-augmented policies
- Multi-task learning to train on multiple environments simultaneously

5) Real-World Industrial Deployment: Ultimate validation requires long-term deployment in actual factory environments:

- Extended field testing (weeks/months of continuous operation)
- Human-robot interaction studies (worker acceptance, safety incidents)
- Economic analysis (cost savings vs. traditional AGVs)

This represents the transition from research contribution to industrial impact.

VIII. Conclusion

This thesis has investigated curriculum learning for sample-efficient deep reinforcement learning in autonomous factory navigation, demonstrating that progressive task decomposition is essential for learning in highly constrained industrial environments. A navigation framework was developed employing Proximal Policy Optimization with multi-modal sensor fusion (camera, ultrasonic, infrared) and a novel three-stage curriculum progressing from empty environments through sparse obstacles to full factory layouts with 0.22-meter-wide corridors. Systematic experimental evaluation across four training paradigms—random baseline, direct training, curriculum learning, and extended training—provides empirical evidence for curriculum learning necessity, improved sample efficiency compared to published work, and critical insights into reward function design.

A. Summary of Contributions

This research contributes preliminary findings to autonomous navigation and reinforcement learning methodology. textbfAll contributions are based on simplified kinematic simulation (MOCK mode) and have not been validated on physical hardware or with realistic sensors:

1) **Curriculum Learning for Factory Navigation:** Designed and evaluated a three-stage curriculum (Empty $\rightarrow$ Sparse $\rightarrow$ Full, 1M total timesteps) that achieves 22% goal-reaching success where direct training fails completely (0% success), showing improvement from complete failure to partial success and providing evidence for the importance of progressive skill acquisition for learning in constrained spaces with sparse rewards.
2) **Sample Efficiency Through Curriculum Learning:** Navigation performance (22% success) was achieved with 1M timesteps compared to published methods [30]–[32] that require 5–15M timesteps for comparable tasks, representing 3–15$\times$ better sample efficiency. Note that direct comparison is approximate as these works address different tasks (multi-robot coordination, maze navigation, outdoor

terrain) with different success criteria and simulation environments. The comparison demonstrates potential benefits of curriculum learning for sample efficiency in navigation tasks generally.

3) **Reward Hacking Analysis and Diagnosis:** Identified, analyzed, and explained a reward hacking failure mode where extended training (5M timesteps) achieved higher reward (+578) but zero success by exploiting dense shaping rewards. Provided mathematical diagnosis demonstrating that accumulated per-step rewards (578) exceeded expected goal-reaching returns (522), causing rational but suboptimal behavior. Proposed corrected reward function with goal dominance guarantee (Eq. 59).
4) **Collision-Free Navigation:** All trained PPO policies achieved 0% collision rate across 100 evaluation episodes, suggesting that curriculum learning and proper reward design can enable simultaneous optimization of task completion (22% success) and safety (0% collisions)—relevant to potential future industrial deployment where equipment protection and worker safety are important.
5) **Reproducible Open-Source Framework:** Fully documented implementation with ROS 2 workspace, Gazebo simulation environments, curriculum learning infrastructure, and evaluation scripts, enabling reproducibility and serving as a foundation for future research in autonomous navigation and curriculum learning methodologies.

B. Key Findings

The experimental evaluation yields the following principal findings:

1) **Curriculum Learning is Essential:** Direct training on the full navigation task (1M timesteps) achieves 0% success despite collision-free navigation, while curriculum training achieves 22% success with identical computational budget. This improvement from complete failure to partial success demonstrates that curriculum learning is not merely beneficial but absolutely necessary for enabling any learning in this task.
2) **Transfer Learning Validation:** Quantitative evidence shows successful skill transfer across curriculum stages: Stage 1 final success (40–50%) $\rightarrow$ Stage 2 initial success (25–35%) $\rightarrow$ Stage 3 initial success ($\sim$15%) $\rightarrow$ Stage 3 final success (22%). The non-zero initial success rates in Stages 2–3 confirm that skills learned in earlier stages transfer to more complex tasks.
3) **Sample Efficiency Advantage:** The proposed approach achieves 22% success at 1M timesteps, comparing favorably with literature results of 10–40% at 5–15M timesteps, representing 3–15$\times$ better sample efficiency. This translates to 2-hour training time vs. 10–30 hours for comparable methods, enabling rapid iteration and reducing energy consumption.
4) **Reward Function Design Principles:** The extended training reward hacking failure empirically validates theoretical principles: (1) terminal rewards must dominate accumulated step rewards (Eq. 59), (2) monitoring reward alone is insufficient—task metrics (success rate, collision rate) are essential, (3) non-potential-based shaping terms can create exploitable local optima violating the potential-based shaping theorem [28].
5) **Safety-Performance Trade-off:** The curriculum model maintains collision-free navigation (0% collisions across 100 episodes) while achieving 22% success, suggesting that well-designed reward functions can enable simultaneous optimization of competing objectives without sacrificing safety for performance gains.
6) **Competitive Absolute Performance:** The 22% success rate is within the range reported in comparable work given task difficulty: 0.22-meter corridors (2–3$\times$ narrower than literature benchmarks), sparse rewards (+200 goal bonus vs. dense waypoint rewards in prior work), and high-dimensional observations (14,400-dimensional camera input). Success rates of 10–40% are standard in comparable navigation research [30]–[33].

C. Practical Implications

This work provides initial evidence toward deploying learned navigation policies on low-cost industrial robots, with implications for:

- **Small and Medium Enterprises (SMEs):** Access to affordable automation without expensive sensors or infrastructure
- **Flexible Manufacturing:** Robots that adapt to layout changes without reprogramming
- **Warehouse Logistics:** Autonomous material transport in dynamic, human-shared environments

D. Limitations and Open Challenges

This research has several critical limitations that significantly constrain the scope and generalizability of findings. These are reiterated here to ensure they are prominently communicated:

Critical Scope Limitations:

1) **Kinematic Simulation Only:** All training and evaluation used MOCK mode—a simplified kinematic simulation without full Gazebo physics. Results do not include friction, inertia, motor dynamics, realistic sensor noise, or visual artifacts.
2) **No Hardware Validation:** The trained policy has not been deployed to physical hardware. Sim-to-real transfer remains completely untested.
3) **Single Environment:** Only one factory layout was tested. Generalization to other environments was not evaluated.
4) **Synthetic Sensors:** Camera observations are procedural gradients, not realistic images. Lidar is simplified ray-casting without noise.
5) **Limited Baselines:** Only random policy comparison. No classical navigation or alternative RL algorithm baselines.

Additional Technical Limitations:

1) **Goal-Reaching Performance:** While curriculum learning achieved 22% success (vs. 0% for direct training), the majority of episodes timeout. The agent shows learned navigation behavior but frequently fails the final approach.

2) **Sample Inefficiency:** On-policy PPO required 1M+ timesteps to achieve modest improvements, corresponding to substantial computational resources (several hours of parallel simulation). Off-policy algorithms or model-based methods may offer improved sample efficiency.
3) **Evaluation Methodology Sensitivity:** The training-evaluation discrepancy reveals critical dependence on evaluation configuration (deterministic policy execution, mock environment approximations, episode length). Future research must employ more comprehensive evaluation protocols including stochastic sampling, full-fidelity simulation, and varied episode configurations.
4) **Sparse Reward Limitations:** The reward function's reliance on sparse terminal rewards ($+200$ goal bonus) created insufficient learning signal when success was initially rare. Progress shaping alone proved insufficient to bridge the exploration gap for distant goals within limited episode horizons.
5) **Sim-to-Real Transfer Not Validated:** All experiments were conducted in simulation; physical deployment on the PiCar-X platform remains future work. Domain randomization strategies were designed but not empirically validated through real-world testing.
6) **Single-Task Specialization:** The learned policies are specialized for point-to-point navigation in factory-like layouts. Generalization to substantially different tasks (multi-goal routing, outdoor navigation, human-robot collaboration) is uncertain without retraining or transfer learning.
7) **Lack of Formal Safety Guarantees:** Neural network policies are inherently black-box, preventing formal verification of safety properties required for certified industrial deployment. The perfect collision avoidance in evaluation provides empirical evidence but not mathematical guarantees.

E. Future Work

The findings of this research motivate several high-priority directions for future investigation:

1) **Curriculum Learning for Goal-Reaching:** Implement progressive goal-distance scheduling, starting with short-distance tasks (2–3 meters) to ensure frequent success and strong value function gradients, then gradually increasing difficulty. This should address the sparse reward problem and enable consistent goal-reaching.
2) **Stochastic Policy Evaluation:** Develop evaluation protocols that sample actions stochastically (rather than deterministically) to assess whether the training-evaluation discrepancy stems from exploration elimination. Compare success rate distributions across sampling modes.
3) **Full-Fidelity Evaluation in Gazebo:** Re-evaluate trained policies in full Gazebo simulation (not mock environment) to isolate the impact of environment approximations on performance. If success rate improves, develop higher-fidelity mock environments or real-time Gazebo evaluation pipelines.
4) **Off-Policy Algorithms (SAC, TD3):** Compare PPO with sample-efficient off-policy methods to reduce training duration and improve data efficiency. Investigate whether replay buffers enable better learning from rare goal-reaching experiences.
5) **Sim-to-Real Transfer Validation:** Deploy trained policies on the physical PiCar-X platform to empirically validate domain randomization effectiveness. Measure sim-to-real performance gaps and iteratively refine randomization distributions.
6) **Hierarchical Policy Architectures:** Separate high-level planning (subgoal selection, route planning) from low-level control (steering, speed regulation) to improve sample efficiency and enable transfer across different factory layouts.
7) **Value Function Convergence Analysis:** Investigate techniques to improve value function learning near the goal region, including prioritized experience replay for near-goal states, auxiliary prediction tasks, or reward bonus scaling for final-approach states.
8) **Formal Verification Integration:** Explore neural network verification tools (e.g., Marabou, α-β-CROWN) to provide probabilistic safety certificates for collision avoidance, potentially enabling certified deployment in controlled industrial settings.

F. Closing Remarks

This thesis provides preliminary evidence that deep reinforcement learning can learn obstacle avoidance and directional navigation in a simplified kinematic simulation of factory environments from synthetic sensory observations, achieving collision-free navigation and 22% goal-reaching success through curriculum learning. While complete goal-reaching execution remains an open challenge, the 68.8% task completion, 16% reward improvement, and 62% variance reduction achieved through systematic training and tuning provide compelling evidence of learning capability.

The research contributes three key insights to the field: (1) reward engineering directly and measurably shapes learned behavior, with coefficient adjustments producing predictable policy changes; (2) evaluation methodology significantly impacts measured performance, with training-evaluation discrepancies revealing sensitivities to sampling mode and environment fidelity; and (3) multi-objective optimization in RL requires careful balance, as overly strong penalties can induce conservative policies that prioritize safety over task completion.

The open-source implementation—built entirely on ROS 2, stable-baselines3, and Python kinematic simulation—provides a reproducible foundation for future research. All results presented are from MOCK mode simulation; validation on physical hardware and full physics simulation remains future work. The documented training procedures, reward function formulations, and evaluation protocols enable direct comparison with alternative approaches and facilitate extension to more complex scenarios.

As reinforcement learning algorithms continue to mature and computational resources become more accessible, the potential exists for learned navigation policies to eventually transition from research demonstrations to deployed industrial systems. The primary remaining barriers—sparse reward learning, sim-to-real transfer validation, and formal safety certification—represent active research frontiers with promising recent advances. This work contributes to that progression by providing empirical evidence, methodological insights, and practical infrastructure for continued investigation toward safe, efficient, and adaptable autonomous factory navigation.

Acknowledgments

The author would like to begin by thanking his family for all the patience, support, and love they have provided throughout his life and for always being there for him.

Gratitude is also extended to the faculty of the Department of Industrial and Manufacturing Engineering at Universidad Autónoma de Ciudad Juárez for their guidance and support.

References

[1] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *International Conference on Machine Learning (ICML)*, 2009, pp. 41–48.

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

[3] I. S. Peyas, A. Sieam, M. N. Islam, and M. A. Islam, "Autonomous warehouse robot using deep q-learning," *arXiv preprint arXiv:2202.05543*, 2022.

[4] Y. Zhu, J. Zhang, C. Li, Y. Li, B. Zhang, and Y. Lu, "Deep reinforcement learning of mobile robot navigation in dynamic environment: A review," *Sensors*, vol. 23, no. 11, p. 5109, 2023.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[6] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[7] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *IEEE International Conference on Robotics and Automation*, vol. 4. IEEE, 1994, pp. 3310–3317.

[8] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," in *Technical Report, Computer Science Department, Iowa State University*, 1998.

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

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[11] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[12] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[13] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ international conference on intelligent robots and systems*, vol. 3. IEEE, 2004, pp. 2149–2154.

[14] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

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

[16] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," in *Machine Learning*, vol. 8, no. 3-4. Springer, 1992, pp. 293–321.

[17] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, vol. 13, 2000, pp. 1057–1063.

[18] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *International Conference on Learning Representations*, 2016.

[19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 1861–1870, arXiv:1801.01290.

[20] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 1587–1596.

[21] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," *arXiv preprint arXiv:1703.06907*, 2017.

[22] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

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

[24] G. Frimodig, "Agv routing on factory floor by reinforcement learning," Master's thesis, Uppsala University, 2024.

[25] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016. [Online]. Available: https://arxiv.org/abs/1606.01540

[26] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. D. Cola, T. Delahaye, R. Goulão, A. Kallinteris, A. KG, M. Krimmel *et al.*, "Gymnasium," https://doi.org/10.5281/zenodo.8127025, 2023.

[27] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[28] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *International Conference on Machine Learning*, 1999, pp. 278–287.

[29] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie Mellon University, Tech. Rep., 1992.

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

[31] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu *et al.*, "Learning to navigate in complex environments," in *International Conference on Learning Representations (ICLR)*, 2017.

[32] Z. Xie and P. Dames, "Learning long-range perception using self-supervision from short-range sensors and odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 420–11 427.

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

[34] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.