GENG5512 MPE Engineering Research Project Part 2

Final Report

Deep Learning for Mobile Robots

Siyi Ma
24533441
School of Engineering, University of Western Australia

Supervisor: Thomas Bräunl
School of Engineering, University of Western Australia

Word count: 5,862

School of Engineering
University of Western Australia

Submitted: 18 May 2026

Project Summary

Deploying autonomous navigation and multi-agent cooperative algorithms on low-cost, compute-constrained edge devices (such as the Raspberry Pi) presents significant engineering challenges. Traditional end-to-end Deep Reinforcement Learning (DRL) models rely heavily on Convolutional Neural Networks (CNNs), which incur severe inference latency and "control lag" on GPU-less hardware. Furthermore, policies trained in pure simulation frequently fail in the physical world due to the "Reality Gap" caused by uneven illumination, as well as the unmodelled mechanical slipping inherent to differential-drive chassis.

To address these interconnected bottlenecks, this thesis proposes an ultra-lightweight, decentralized visual reinforcement learning framework tailored for the EyeBot teaching and research platform. To fundamentally bypass the computational expense of 2D convolutions, the system innovatively employs a 1D grayscale projection dimensionality reduction pipeline coupled with a lightweight Multi-Layer Perceptron (MLP) trained via Proximal Policy Optimization (PPO). To overcome unpredictable physical lighting interference, a synergistic CV-RL co-design is introduced: combining robust onboard OpenCV preprocessing (incorporating CLAHE and Adaptive Thresholding) with aggressive visual Domain Randomization in the simulator. Additionally, a structured composite reward function featuring a novel kinematic difference penalty is formulated to guide the agent in autonomously respecting physical friction limits.

Real-world physical experiments demonstrate that the proposed minimalist architecture compresses the single-frame end-to-end inference latency to under 50 ms (20+ FPS), thoroughly unlocking the physical speed limits of the hardware. Relying exclusively on onboard inference, the system flawlessly achieves "Zero-Shot Transfer" from simulation to the physical track without any weight fine-tuning. The agent successfully exhibits high-speed agile tracking, completely eradicates uncontrolled sideslipping during sharp cornering, and maintains a strict zero-collision rate during dynamic dual-vehicle cooperative following. Ultimately, this research provides a highly efficient, physics-compliant, and scalable architectural paradigm for edge AI robotics deployment.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Dr. Thomas Bräunl, for his invaluable guidance, continuous support, and inspiring vision throughout this project.

I am also deeply grateful to Prof. Norbert Oswald for his profound mentorship and insightful advice, which have greatly shaped my academic journey.

Special thanks go to Kieran Quirke-Brown; his engaging and comprehensive teachings in Embedded Systems and Mobile Robots laid a solid technical foundation for my research.

I would also like to extend my sincere appreciation to my seniors, Noah Mueller and Ben Nicholson, for their generous help, technical assistance, and willingness to share their expertise in the lab.

Finally, none of this would have been possible without the unconditional love, patience, and unwavering support of my family and friends. Thank you for always standing by my side and believing in me.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

In the fields of mobile robotics and autonomous driving, enabling decentralized, collision-free coordination among multiple independent agents remains a fundamental engineering pursuit. To systematically study and evaluate these complex behaviours, academia frequently relies on scaled-down laboratory testbeds featuring structured lane networks—commonly abstracted as high-contrast line-following tracks. By deliberately filtering out unstructured environmental noise, these environments compel miniature robots to rely entirely on constrained onboard perception to negotiate track topologies, identify intersections, and dynamically maintain safe following distances.

Concurrently, the paradigm of autonomous navigation has increasingly shifted towards Deep Reinforcement Learning (DRL). Unlike traditional rule-based Computer Vision (CV) pipelines, DRL offers an end-to-end approach, allowing agents to map raw sensory inputs directly to continuous control commands through trial and error. Consequently, integrating these advanced AI models with low-cost, edge-computing educational platforms—such as the Raspberry Pi-driven EyeBot [1]—has emerged as a crucial initiative. This integration not only democratizes advanced robotics research but also provides an accessible testbed for validating multi-agent cooperative algorithms.

However, the transition of DRL-based navigation from theoretical simulation into physical edge deployment is far from trivial. When bringing these theoretical models into the real world, researchers are immediately confronted by the harsh physical and computational limitations of edge devices.

1.2 The Issue and Challenges

When migrating the visual reinforcement learning algorithm from the ideal simulator (EyeSim [2]) to the affordable physical EyeBot robots, this study encountered three specific engineering bottlenecks:

- **Severe Computational Constraints of Edge Hardware:** Traditional end-to-end visual models rely on deep convolutional neural networks (CNNs) [3]. Deploying these on a GPU-less Raspberry Pi incurs severe inference latency, resulting in delayed responses when the robot engages in multi-vehicle following or encounters sharp turns, thereby significantly increasing the risk of collisions.
- **Sim-to-Real Vision Gap:** Real-world physical laboratories inevitably contain window glare and overhead lighting shadows. Such uneven illumination easily disrupts the binarized features of the track, causing models that perform perfectly in simulation to deviate from the track during physical testing.
- **Physical Chassis Slip and Oscillatory Behaviour:** Real differential-drive robots are constrained by physical friction during operation. Due to the mechanical transmission characteristics of the chassis, if the angular velocity output by the controller is significantly greater than the linear velocity, the drastic rotational speed difference between the left and right wheels generates excessive instantaneous torque. This causes the tires to immediately overcome the ground's static friction, resulting in uncontrolled slipping or fishtailing.

Furthermore, traditional RL policies lack action smoothness constraints, which frequently leads to high-frequency lateral oscillations (weaving) along straight track sections.

1.3 Project Objectives

The primary objective of this project is to develop and deploy a decentralized, ultra-lightweight visual reinforcement learning navigation system tailored for highly compute-constrained edge teaching and research platforms (specifically, the Raspberry Pi-powered EyeBot-8 robots). This system is designed to enable multiple independent miniature robots to achieve smooth, stable path following and autonomous, collision-free coordination on standardized laboratory line-following tracks (comprising straight segments, curves, and intersections), relying exclusively on their own low-latency onboard inference.

To achieve the aforementioned primary objective, this project establishes the following three specific and measurable engineering sub-objectives:

- Extreme Model Compression for Ultra-Low Latency Inference. To overcome the computational bottlenecks of the GPU-less Raspberry Pi, the primary objective is to design a highly compressed state representation method, completely eliminating the reliance on massive Convolutional Neural Networks (CNNs). The measurable target is to achieve an end-to-end single-frame inference latency of less than 50ms, thereby thoroughly resolving the control lag issue and ensuring real-time, agile control of the physical robot.
- Robust Illumination Compensation for Zero-Shot Transfer. To address unpredictable lighting interference (such as glare and local shadows) in the physical laboratory, the second objective is to develop a low-computational-cost visual preprocessing pipeline. The measurable target is to completely bridge the visual domain gap, enabling the policy trained in the virtual environment to be robustly deployed onto the physical EyeBot robots without requiring any secondary training or weight fine-tuning in the real world (achieving Zero-Shot Transfer).
- End-to-End Learning of Physical and Safety Constraints. To replace traditional hardcoded anti-slip and braking rules in the underlying code, the third objective is to guide the neural network to autonomously perceive and respect the physical kinematic limits of the chassis and safe following distances through Reward Shaping [4]. The measurable targets are to completely eliminate high-frequency chassis oscillations (weaving) and uncontrolled slipping during cornering, and to achieve a strict 0-collision rate during dynamic multi-agent following tasks.

2. Literature Review

This chapter establishes the hardware context and reviews the technological evolution of autonomous navigation. By analysing the kinematics of the chosen robotic platform, the limitations of traditional computer vision, the computational bottlenecks of neural networks on edge devices, and the challenges of Sim-to-Real transfer, this chapter identifies the specific research gaps that this thesis aims to address.

This research is grounded in the University of Western Australia's EyeBot teaching and research platform. The robot is powered by a Raspberry Pi CPU, devoid of a dedicated Graphics Processing Unit (GPU), making computational efficiency a paramount constraint. The physical robot employs a differential-drive chassis, where navigation is governed by adjusting the rotational speeds of the left wheel ($\dot{\theta}_L$) and the right wheel ($\dot{\theta}_R$). Based on differential drive inverse kinematics, the relationship between the robot's target linear velocity (v), target angular velocity (ω) , the wheel radius (r), and the required individual wheel speeds (in revolutions per second) is defined by the following equations:

$$\dot{\theta}_L = \frac{1}{2\pi r}(v - \frac{\omega d}{2}) \tag{2.1}$$

$$\dot{\theta}_R = \frac{1}{2\pi r}(v + \frac{\omega d}{2}) \tag{2.2}$$

(where d represents the wheelbase). From a mechanical perspective, these equations reveal a critical vulnerability: if a control algorithm commands an extremely large angular velocity (ω) while the forward linear velocity (v) is close to zero, the wheels will be forced to spin at high speeds in opposite directions. The excessive instantaneous torque generated by this speed difference easily overcomes the static friction of the ground, causing the robot to slip or spin out of control. Understanding this physical constraint is vital for designing safe autonomous systems.

Historically, before the widespread adoption of Deep Learning, autonomous navigation on such edge devices heavily relied on traditional Computer Vision (CV) pipelines. For instance, previous research utilizing an earlier generation of the EyeBot platform achieved lane following and traffic sign recognition by combining Canny edge detection, Hough transforms (see Figure 2.1), and Support Vector Machines (SVM) [5]. However, this traditional paradigm exhibits several inherent engineering flaws. Firstly, supervised feature classification (such as training an SVM) inherently requires massive amounts of manual image annotation, which is highly labour-intensive and poorly scalable. Secondly, these classical systems typically rely on rigid, rule-based logic to process geometrical distances, lacking the ability to learn smooth, continuous control end-to-end. Most importantly, despite not using deep neural networks, the dense pixel-by-pixel calculations severely bottleneck the CPU. As explicitly reported by Sun et al. [5], the main control loop for their traditional pipeline on the EyeBot averaged an execution time of 340 ms per frame (less than 3 FPS). This fatal "control lag" renders high-speed dynamic tracking physically impossible.

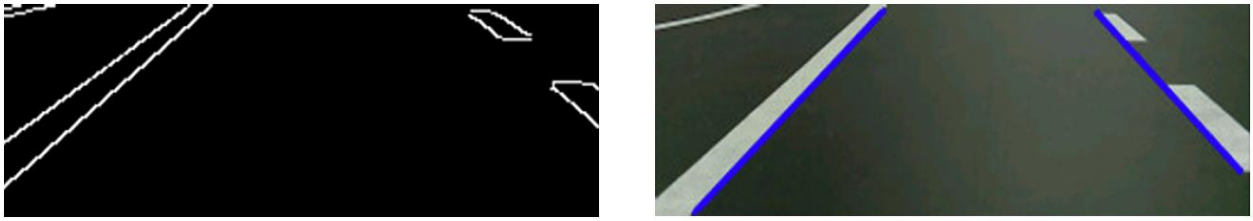

Figure 2.1: The traditional computer vision pipeline for lane detection on the EyeBot platform. The process heavily relies on dense pixel-wise operations, such as Canny edge detection (left) and rigid geometric line fitting via Hough transforms (right). Image source: Sun et al. [5].

To overcome the limitations of manual feature engineering and discrete rule-based control, end-to-end Deep Reinforcement Learning (DRL) has emerged as a dominant solution for various autonomous navigation tasks [6]. DRL algorithms, particularly Proximal Policy Optimization (PPO) [7], automatically learn optimal policies through trial and error, directly mapping raw sensory inputs to continuous steering and acceleration outputs. In pure virtual simulation environments, PPO can quickly converge and perform flawless autonomous navigation. However, deploying standard DRL architectures onto the physical EyeBot exposes a critical computational bottleneck. Standard visual DRL models rely on Convolutional Neural Networks (CNNs) to extract spatial features from 2D image matrices [3]. Processing a single frame through a standard CNN on a Raspberry Pi often incurs latency exceeding 150 milliseconds. Previous attempts to accelerate neural networks on edge devices primarily focused on network quantization or pruning [8]. Yet, these methods only offer marginal improvements and fail to eliminate the fundamental computational expense of 2D sliding-window convolutions. Consequently, a critical architectural gap remains in the literature: rather than marginally compressing CNNs, how to effectively couple deterministic 1D visual projection techniques with lightweight Multi-Layer Perceptron (MLP) within a Reinforcement Learning framework to entirely bypass 2D convolutions. Such a hybrid architectural paradigm for compute-constrained edge devices remains underexplored.

Furthermore, even if this computational bottleneck is resolved, RL policies trained in perfect simulators consistently fail in the physical world due to unmodelled environmental noise—a phenomenon known as the "Reality Gap" [9]. Simulations provide pristine track lines and uniform lighting, whereas physical laboratories suffer from severe diffuse reflections, window glare, and local shadows. To bridge this gap, researchers commonly employ Domain Adaptation (using GANs to generate photo-realistic images) [10] or Domain Randomization (intentionally injecting massive random noise during training, as illustrated in Figure 2.2) [11]. While Domain Randomization is highly effective and computationally free during inference, extreme physical lighting often exceeds the distribution of randomized training data. Therefore, an intermediate, ultra-lightweight visual preprocessing step is required to normalize the physical input before feeding it to the network. The synergistic co-design of lightweight CV preprocessing on the real robot and Domain Randomization in the simulator remains an underexplored solution for bridging the reality gap on compute-constrained platforms.

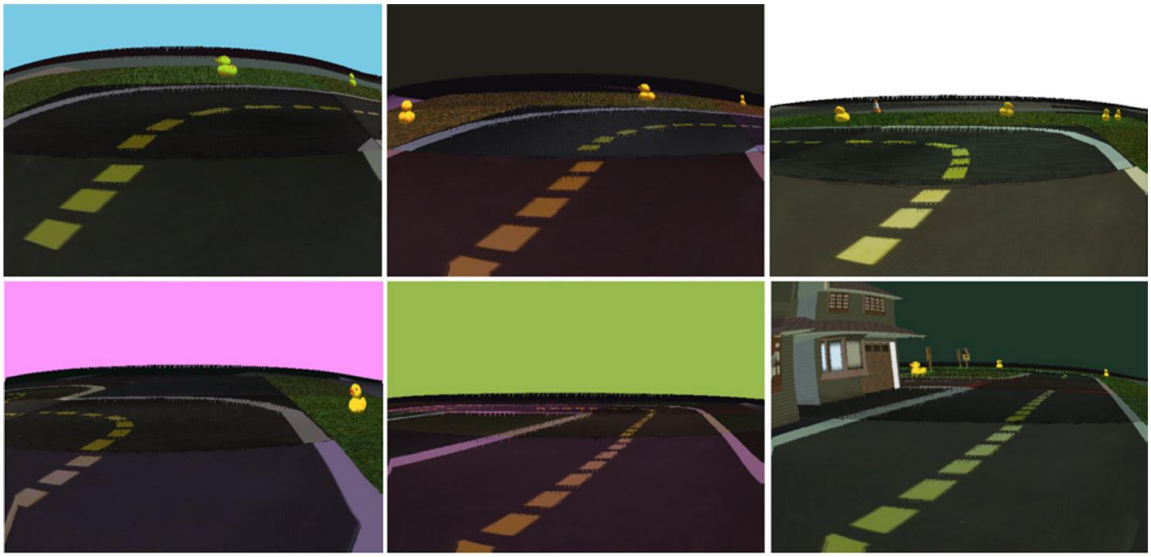

Figure 2.2: Examples of visual Domain Randomization. Extreme variations in lighting and textures are injected during training to bridge the "Reality Gap". Image source: Béres and Gyires-Tóth [12].

Consequently, a critical review of the existing literature reveals three distinct research gaps: the lack of a hybrid architectural paradigm to bypass CNN latency on GPU-less CPUs; the insufficient integration of kinematic constraints leading to uncontrollable chassis slipping; and the need for a low-compute Sim-to-Real transfer methodology. Addressing these interconnected challenges forms the core motivation of this thesis.

3. Project Process and Experimental Design

This chapter provides a detailed description of the project's hardware and simulation architectures, the design of the ultra-lightweight dimensionality reduction feature pipeline, the mathematical formulation of the reinforcement learning algorithm and composite reward function, and the real-world visual preprocessing pipeline aimed at overcoming the "Reality Gap". This systematic experimental design ensures both the computational efficiency of the algorithm on edge devices and the robustness of its physical deployment.

3.1 Hardware and Simulation Platforms

This study employs the University of Western Australia's standard EyeBot-8 robot as the compute-constrained edge teaching and research platform. Low-level motor driving and PSD sensor data acquisition are handled by a custom EyeBot I/O expansion board, while high-level visual data preprocessing (based on an original resolution of 320×240) and neural network inference are executed by the compute-constrained central processing unit of the Raspberry Pi. Model training is conducted in the proprietary 3D robot simulator, EyeSim, where a standardized black-and-white line-following track comprising straight segments, curves, and intersections was constructed (as illustrated in Figure 3.1).

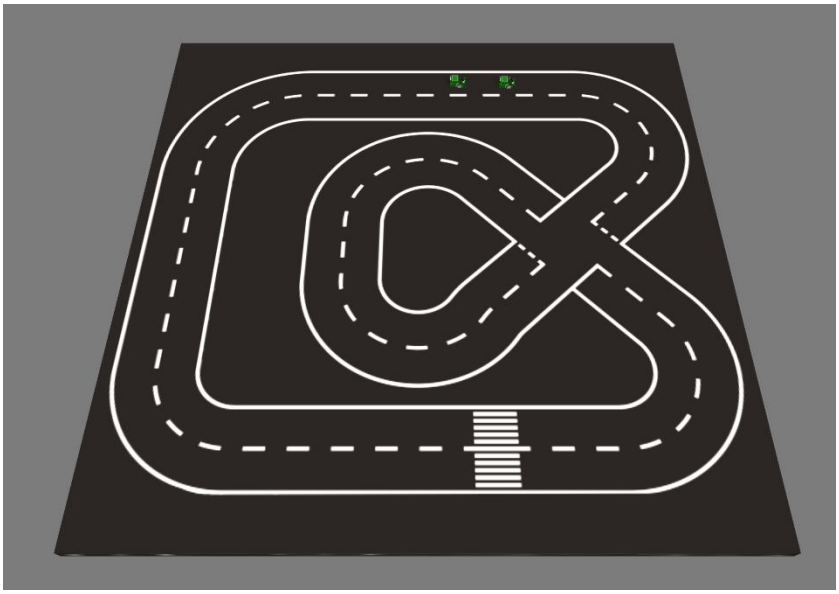

Figure 3.1: Overview of the EyeSim 3D simulation environment used for policy training, featuring a standardized track layout with an intersection.

To prevent the policy network from overfitting to a single following speed and to compel it to learn highly generalizable adaptive cruise control and emergency obstacle avoidance logic, this study introduces a deep Environment Randomization mechanism [13] for the leading vehicle in the simulation environment. During the training process, the leading vehicle is programmed to dynamically switch among four motion modes based on specific probabilities: Fast (10%), Normal (30%), Slow (30%), and Stop (30%). The duration of each mode fluctuates randomly between 1.0 and 3.5 seconds. Furthermore, at each episode reset, the leading vehicle is randomly spawned 2 to 4

path points (grids) ahead of the agent. This highly dynamic and unpredictable adversarial configuration ensures that the agent can adeptly handle extreme and sudden situations in the physical world, such as emergency braking by the leading vehicle.

3.2 Ultra-Lightweight State Space Formulation

To achieve ultra-low latency real-time control on the GPU-less Raspberry Pi, this study completely discards computationally expensive 2D convolution operations. Instead, it develops a feature dimensionality reduction pipeline based on 1D grayscale projection to construct a highly compact Markov Decision Process (MDP) state space.

This dimensionality reduction pipeline comprises the following key steps:

1. **ROI Extraction:** Upon acquiring the grayscale image from the front-facing camera, the system directly discards the upper half of the field of view (which contains irrelevant background elements such as walls and the horizon). It only crops the lower half of the image (320×120 pixels) as the processing core, thereby forcing the network's attention to strictly focus on the drivable track surface.
2. **1D Grayscale Projection & Normalization:** The ROI is vertically divided into three equal zones: Top/Far, Mid, and Bot/Near (each 320×40 pixels). For each of these three zones, pixel column sums are calculated along the vertical dimension, and these values are strictly normalized to the [0, 1] interval by dividing them by the theoretical maximum integral value (255×40). This process generates a 960-dimensional feature vector (as prominently illustrated in Figure 3.1) that accurately captures track edges and curvature. Simultaneously, row sums are calculated along the horizontal dimension for the bottommost Near zone and subsequently normalized, generating a 40-dimensional feature vector utilized for detecting starting lines or intersections ahead.
3. **Sensor Fusion and Frame Stacking:** The distance reading from the front-facing PSD sensor is acquired and normalized to [0, 1]. To prevent this single distance scalar from being overshadowed by the massive visual features within the network (gradient dilution), the system strategically duplicates this PSD scalar 40 times and concatenates it to the visual features. This enhances the weighting of distance perception, resulting in a 1,040-dimensional reduced vector per frame. Finally, to equip the system with the capability to perceive temporal context (such as relative speed and acceleration), the most recent 4 frames of data are stacked (1,040×4) [3], ultimately constructing the 4,160-dimensional network input state.

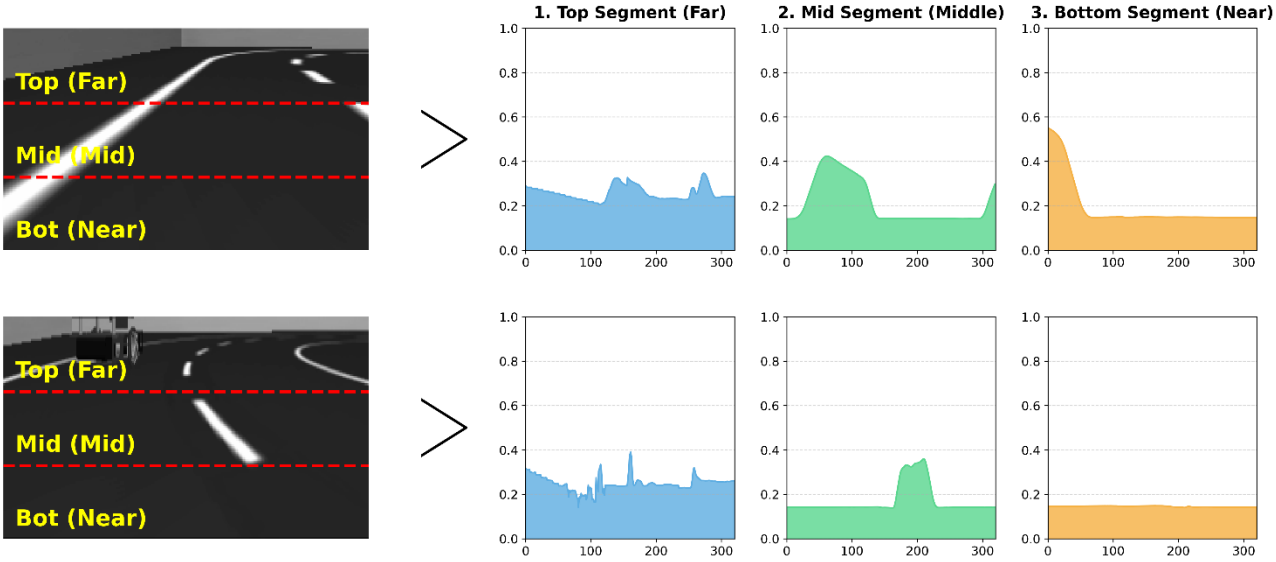

Figure 3.2: Illustration of the 1D Grayscale Projection pipeline, converting the visual ROI into 1D feature waveforms.

3.3 RL Algorithm and Composite Reward Shaping

Benefiting from the extreme compression of the state space into a 1D vector, the system successfully replaces the CNN architecture with a lightweight Multi-Layer Perceptron (MLP policy), utilizing the Proximal Policy Optimization (PPO) [7] algorithm for policy iteration. This architectural paradigm shift perfectly aligns with the stringent low-memory and ultra-low latency requirements of edge devices.

To guide the agent in learning flawless coordination and safe driving amidst complex physical dynamics, this study designs a highly structured Composite Reward Function. At each time step t, the total reward R consists of multiple weighted components, governed by the following core mathematical framework:

$$R = \alpha R_{progress} + \gamma R_{speed} + \beta R_{time} + \epsilon R_{following} + \delta R_{smoothness} + \theta R_{kinematic} + \eta R_{penalty} \quad (3.1)$$

The specific physical significance and mathematical formulation of each component are detailed as follows:

3.3.1 Kinematics & Efficiency Guidance

- Progress Reward ($R_{progress}$): Employs potential-based reward shaping [14], measuring effective displacement by calculating the difference in distance to the target path point (centroid) between the previous and current frames: $R_{progress} = (d_{t-1} - d_t)$. This mechanism ensures the agent is rewarded only when it genuinely reduces the distance to the target, effectively preventing the policy from "farming rewards" by staying stationary.
- Speed and Time Incentives (R_{speed} & R_{time}): Assigns a positive reward to the current linear velocity action A_{linear} to encourage efficient driving, while imposing a constant time penalty to discourage idling or spinning in place: $R_{speed} = A_{linear}$, $R_{time} = -1$.

3.3.2 Cooperative Following & Predictability

To achieve smooth multi-agent following, the system extracts the absolute coordinates of both vehicles directly from the underlying simulation engine during the training phase. It computes the precise relative Euclidean Distance (D_{lead}) as the "Ground Truth" for reward evaluation. The system implements a piecewise adaptive reward ($R_{following}$), with thresholds defined in Simulation Units (SU):

- Sweet Spot: When $500 \leq D_{lead} \leq 900$, a maximum stable reward of $+0.1$ is granted.
- Exponential Decay (Catching up): When $D_{lead} > 900$, an exponential decay function smoothly guides acceleration $R_{following} = 0.1 \times exp\left(-\frac{D_{lead}-900}{1000}\right)$.
- Linear Penalty (Approaching warning): When $300 < D_{lead} \leq 500$, a proportional linear negative penalty is applied, compelling the network to decelerate proactively.

Here, the system deliberately decouples the reward evaluation from the observational input of the front-facing PSD sensor. This design effectively circumvents the classic "Sensor Aliasing" problem: because the PSD cannot physically distinguish whether the object ahead is the "leading vehicle" or the "wall at the track's edge," using it directly for penalty calculations would cause the vehicle to frequently trigger "False Positive" collision penalties when sweeping past walls during cornering, leading to irrational emergency braking. By relying on absolute coordinates for reward calculation, the system ensures the network remains focused purely on the cooperative following objective, eliminating interference from environmental spatial artifacts.

3.3.3 End-to-End Physical Constraints

The system completely discards underlying hardcoded anti-slip rules, relying entirely on the following constraints to allow the network to autonomously learn physical intuition and chassis limits:

- Action Smoothness Penalty ($R_{smoothness}$): Introduces an L1 norm penalty on the difference between the current and previous frame's actions to suppress high-frequency oscillatory commands, thereby providing a highly predictable trajectory for the following vehicle: $R\_smoothness = \sum |A_t - A_{t-1}|$.
- Kinematic Anti-Slip Penalty ($R_{kinematic}$): Tailored to the physical characteristics of a real differential-drive chassis. Empirical testing reveals that when the output angular velocity significantly exceeds the linear velocity, the wheels easily lose grip and slip due to excessive instantaneous torque. To address this, the system introduces a difference penalty term targeting the "portion of angular velocity that exceeds linear velocity": $R\_kinematic = max\left(0, |A_{angular}| - A_{linear}\right)$.
- Terminal Penalty ($R_{penalty}$): Encompasses discrete, event-driven penalties. When the distance falls below the absolute safety threshold ($D_{lead} < 300$), a severe collision penalty is triggered and the episode is reset (facilitating end-to-end learning of emergency braking); when the vehicle deviates off-track on both sides, a heavy out-of-bounds penalty is triggered; conversely, successfully passing a waypoint yields a substantial one-time reward.

Following extensive simulation iterations, the weights of the reward components and key hyperparameters that ultimately achieved successful convergence and smooth deployment to the physical robot are detailed in Table 3.1.

Table 3.1: Empirical Weights and Parameters for the Composite Reward Function

Parameter	Description	Value
α	Weight for Progress Reward ($R_{progress}$)	0.02
γ	Weight for Speed Incentive (R_{speed})	0.02
β	Weight for Time Penalty (R_{time})	0.01
ϵ	Max Weight for Distance Keeping ($R_{following}$)	0.1
δ	Weight for Smoothness Penalty ($R_{smoothness}$)	-0.1
θ	Weight for Kinematic Constraint ($R_{kinematic}$)	-0.05
$\eta_{collide}$	Emergency Braking / Collision Penalty	-20.0
$\eta_{offtrack}$	Off-track Penalty	-10.0

3.4 Zero-Shot Sim-to-Real Preprocessing

To overcome the complex and variable lighting disturbances in the physical laboratory (e.g., window glare and overhead light shadows) and achieve "Zero-Shot Transfer" without altering any model weights, this study designs and deploys a robust OpenCV-based Image Preprocessing Pipeline on the physical robot for inference.

Because lighting in the simulation environment is absolutely uniform, whereas the real track suffers from severe diffuse reflections and local shadows, directly inputting physical images would cause the model to fail catastrophically. Therefore, the system executes the following four rigorous feature-cleaning steps on the raw images captured by the physical camera:

1. **Contrast Limited Adaptive Histogram Equalization (CLAHE)** [15]**:** Discarding global histogram equalization, which frequently causes large-area overexposure, the system adopts CLAHE. By dividing the image into an 8×8 grid for local processing and setting clipLimit=2.0 to suppress noise amplification in dark background areas, the system successfully pulls the highly uneven spatial lighting back to a unified baseline.
2. **High-Frequency Noise Suppression (Gaussian Blur):** After contrast stretching, the base noise of the physical camera is slightly amplified. The system introduces a Gaussian Kernel to smooth the image, effectively eliminating the high-frequency electrical noise from the sensor.
3. **Dynamic Feature Extraction (Adaptive Gaussian Thresholding):** To accurately extract track white lines even at the boundaries of strong reflections and deep shadows, the system utilizes adaptive Gaussian thresholding. This algorithm dynamically calculates the threshold based on the weighted average of the pixel's local neighbourhood, achieving robust track contour extraction.
4. **Morphological Cleaning (Morphological Operations):** Finally, to address the residual tiny floor reflections (white dots) and shadow-induced defects on the white lines (black holes)

after binarization, the system sequentially executes morphological OPEN and CLOSE operations, thoroughly smoothing out the minor imperfections of the physical world.

As illustrated in Figure 3.2, the physical images cleaned through this pipeline possess track features with a purity that extremely closely approximates the ideal simulation environment. The processed images are then fed into the 1D projection pipeline for dimensionality reduction and inference, perfectly bridging the "Reality Gap."

Figure 3.3: The Sim-to-Real visual preprocessing, transforming the raw physical feed (left) into clean track features (middle) that highly resemble the ideal simulation environment (right).

It is worth noting that although the aforementioned 4-step OpenCV preprocessing pipeline greatly purifies the reality images, extremely strong local diffuse reflections in the actual physical environment will still leave a small amount of irremovable residual white spots in the final binarized images. To compensate for the physical limitations of classical computer vision processing, this study deliberately injects Random White Noise into the perfect virtual track images during the training phase in the pure virtual simulation (EyeSim). This strategy of proactively "adding noise" on the training end perfectly simulates the reflective artifacts that remain after OpenCV processing during real-robot deployment, forcing the neural network (MLP) to learn to ignore these local feature glitches during training. Through this synergistic CV-RL Co-design of "maximizing denoising on the reality end, and proactively adding noise on the simulation end," the system achieves a perfect alignment of feature distributions. This significantly enhances the generalization capability of the policy network in unseen physical environments, fundamentally ensuring the robust deployment of Zero-Shot Transfer on the actual robot.

3.5 Real-World Experimental Method & Deployment

The PPO policy, trained entirely within the pure virtual simulation, is exported to the ONNX (Open Neural Network Exchange) format. It is subsequently deployed on the physical EyeBot-8's Raspberry Pi using the onnxruntime engine to execute closed-loop inference. Crucially, the final exported inference model occupies a minimal footprint of approximately 2MB. This extreme compactness perfectly satisfies the stringent low-memory and ultra-low latency requirements inherent to edge computing devices, enabling seamless real-time control on the physical robot.

4. Results and Discussion

This chapter presents a comprehensive quantitative and qualitative evaluation of the proposed lightweight visual reinforcement learning system. The evaluation focuses on comparing computational efficiency on edge devices, conducting ablation studies on the preprocessing pipeline, and analysing chassis dynamic performance based on physical constraints. Finally, a critical discussion on the physical and perceptual limitations of the current system is provided.

4.1 Computational Efficiency and Edge Inference

To validate the computational advantages of the proposed system on edge devices, this study conducted a real-world performance comparison on the GPU-less physical Raspberry Pi platform, evaluating the proposed "1D Projection + MLP" architecture against a traditional end-to-end visual reinforcement learning baseline. As the baseline, the system utilized the default CnnPolicy from the Stable Baselines3 framework (i.e., the classic Nature-CNN architecture comprising three convolutional layers), taking a 320×120 2D image matrix as input.

A critical engineering pain point was identified during real-world testing: because the CNN architecture introduces severe inference latency on the Raspberry Pi (single-frame processing typically >150 ms), the system experienced fatal "Control Lag." To prevent the robot from veering off the track due to an inability to steer in time between frames, when deploying the CNN model, it was necessary to artificially and severely restrict the maximum linear and angular velocities of the chassis. This forced the robot to crawl at an extremely sluggish pace, completely sacrificing dynamic agility.

In stark contrast, the proposed system entirely eliminates 2D convolutions via the 1D grayscale projection. As demonstrated in Table 4.1, the final exported ONNX model has a footprint of only about 2MB, and the single-frame end-to-end inference latency is strictly compressed to under 50 ms (achieving real-time control at 20+ FPS). This ultra-low latency completely unlocks the vehicle's physical speed limits, empowering the agent to maintain high driving speeds while still possessing ample reaction time to negotiate sharp turns and sudden braking by a leading vehicle.

Table 4.1: Comparison of Computational Efficiency and Physical Speed Limits on the Raspberry Pi Platform

Architecture	State Space	Inference Latency	FPS	Physical Performance
Traditional Baseline (SB3 CnnPolicy)	320×120 2D Image	> 150 ms	3-6 FPS	Maximum speed severely restricted; extremely sluggish driving
Proposed (1D Projection + MLP)	4,160-D 1D Vector	< 50 ms	20+ FPS	Unlocks hardware speed limits; achieves high-speed agile tracking

4.2 Zero-Shot Transfer and Ablation Study

To validate the necessity of the image preprocessing pipeline proposed in Section 3.4 during real-world deployment, this study conducted an ablation study on the actual laboratory track (illustrated in Figure 4.1), which was subjected to strong sidelight interference (glass wall glare) and diffuse reflections from overhead lighting. The system evaluated the performance differences among the following visual processing strategies:

1. **Raw Image Baseline (No Processing):** Grayscale images from the physical camera were fed directly into the network. Due to the presence of substantial environmental base noise and uneven illumination, the 1D projection features collapsed completely. The agent lost its sense of direction instantly upon starting (survival time < 2 seconds), resulting in a catastrophic derailment.
2. **Pure Global Thresholding:** During early testing, attempts were made to calculate the average brightness of the entire image or to set a fixed numerical value as the sole global threshold for binarization. Empirical testing revealed that this approach fails catastrophically in environments with extreme uneven lighting. The core issue lies in the fact that, in the physical world, the absolute pixel values of "white lines in shadows" and "black floors in highlighted areas" overlap severely (the grayscale value of a white line in a shadow can even be lower than the grayscale value of a black floor under a highlight). Therefore, no matter where this single global threshold is set, it faces an inescapable dilemma: either the track in the shadow areas disappears completely (misclassified as black), or the floor in the highlighted areas turns largely white (generating massive false edges). In actual deployment, this strategy could only manage to travel a short distance on specific starting segments that coincidentally matched its preset threshold. As the vehicle progressed along the track, natural variations in ambient lighting caused the absolute pixel values to rapidly deviate from the fixed threshold, leading the robot to be frequently misled by erroneous features and inevitably derail.
3. **Full Proposed Pipeline:** To address the aforementioned physical pain point of "absolute numerical overlap," the final comprehensive pipeline of this system first utilizes CLAHE to level the global baseline of local contrast. It then enforces Gaussian blur to smooth out physical high-frequency base noise and tiny reflections. Based on this, it executes Adaptive Gaussian Thresholding segmentation, and finally applies morphological operations. This combined approach successfully suppresses the outbreak of local noise and extracts extremely pure track contours.

Relying on this rigorous preprocessing pipeline, coupled with the visual domain randomization (noise injection) strategy introduced during training, the physical robot successfully accommodated the extremely minor residual reflections in the real-world environment without any real-world fine-tuning (Zero-Shot Transfer). It achieved continuous, high-speed, and smooth driving for multiple laps on the real track (with 0 manual interventions for derailment throughout the entire process). This striking ablation comparison compellingly demonstrates that a single computer vision algorithm cannot bridge the complex "Reality Gap" on its own; the synergistic design of multi-step feature cleaning and reinforcement learning domain randomization is absolutely indispensable.

Furthermore, by comparing the physical testing track (Figure 4.1) with the training environment (previously shown in Figure 3.1), it is evident that the physical testbed possesses a completely

different topological layout. The agent's successful navigation on this unseen track powerfully demonstrates robust spatial generalization, proving that the policy learned universal driving behaviors rather than merely memorizing a specific map.

Figure 4.1: The physical laboratory track and the EyeBot robot used for real-world testing, exhibiting visible surface glare and lighting interference.

4.3 Kinematic Constraints and Cooperative Following

This section quantitatively evaluates the agent's chassis control and safe following capabilities in a complex multi-vehicle environment.

4.3.1 Mitigation of Uncontrollable Slipping

During early real-world testing, if the maximum speed limit was not artificially locked in the underlying code, traditional RL policies tended to output extremely radical action commands when facing sharp turns to pursue the fastest heading angle correction—specifically, outputting a massive steering angular velocity while having almost no forward linear velocity. Due to the mechanical transmission characteristics of a differential-drive chassis, this command where "linear velocity is far lower than angular velocity" causes an extreme reverse rotational speed difference between the left and right wheels. This instantaneously breaks the static friction of the ground, manifesting as the vehicle losing forward grip, continuously slipping in place, and spinning wildly out of control. After introducing the kinematic difference penalty ($R_{kinematic}$), the model learned real physical laws from an end-to-end perspective: the agent autonomously realized that "massive steering magnitude must rely on sufficient forward velocity to maintain grip." Empirical data indicates that, without locking the underlying speed, the baseline model lacking this penalty had a starting failure rate of 100%. The vehicle would fall directly into a deadlock state of frantic spinning in place at the very instant of starting or upon encountering the first curve (with an actual effective travel distance close to 0). Conversely, after incorporating the penalty, the vehicle completely overcame the issues of in-place sharp turning and sideslipping. The agent learned to carve smooth cornering arcs while maintaining

forward speed, keeping 0 sideslip throughout the journey. The variance of action commands was significantly reduced, fundamentally resolving the uncontrollable physical slipping phenomenon.

4.3.2 Real-World Cooperative Following and AEB

To validate the dynamic following policy on the physical track, this study deployed two physical robots, equipped with the exact same RL inference model, for cooperative testing. To specifically evaluate the trailing vehicle's deceleration and hazard avoidance capabilities, the maximum speed of the leading vehicle was artificially reduced during specific testing segments. On-site observations demonstrated that when the trailing vehicle approached the slow-moving leading vehicle at high speed, the system could instantly trigger precise deceleration or braking commands (Automatic Emergency Braking, AEB) to lock in a safe following distance. During the conventional constant-speed following phase, the vehicle could smoothly fine-tune its linear velocity, effectively avoiding the stop-and-go oscillatory braking common under traditional rule-based control. Furthermore, when entering structurally complex intersection areas, the agent similarly exhibited extremely reliable cooperative capabilities. Despite the drastic changes in visual features at the intersection and the convergence of the two vehicles' trajectories, the physical system robustly achieved safe passage with 0 collisions across all sections.

4.4 Limitations and Critical Discussion

While the system demonstrates outstanding performance in standardized tests, it still exhibits inherent limitations under extreme physical conditions and specific sensor constraints, illuminating directions for future improvements:

- **Spatial and Semantic Feature Loss in 1D Projection:** Although the 1D grayscale projection extracts track edges and curvature with minimal computational overhead, it fundamentally sacrifices the 2D spatial structure in the vertical dimension. Consequently, the system can only recognize longitudinally extending "line" distributions and cannot comprehend complex graphical or textual semantics. For instance, if a traffic sign with specific shapes or lettering (e.g., a "STOP" sign) appears on the track surface ahead, the current visual pipeline loses all morphological features upon 1D compression, rendering the system incapable of high-level semantic command recognition. To process such traffic indicators in the future, an auxiliary 2D image classification branch must be integrated.
- **PSD Narrow FOV and Perception Bias:** Although the system's dynamic following policy relies on the joint input of 1D visual features and PSD data, the network's judgment of absolute distance becomes highly reliant on the single front-facing PSD sensor during training. Due to the extremely narrow detection beam of this sensor (akin to single-point ray ranging), its effective Field of View (FOV) is severely restricted. When navigating sharp curves, the leading vehicle can easily deviate briefly from the trailing vehicle's direct longitudinal axis, thereby slipping out of the PSD's detection ray. At this juncture, even if the visual input still contains partial contours of the leading vehicle, the dominant PSD ranging reading experiences an instantaneous anomaly (falsely reporting a clear path ahead). The superposition of this physical geometric blind spot and the neural network's perception dependency bias creates the illusion of an "open road," which may trigger a momentary and unintended acceleration.

5. Conclusions and Future Work

5.1 Research Summary

Addressing the computational bottlenecks and physical transfer challenges faced by edge computing devices (e.g., Raspberry Pi) in autonomous driving and multi-agent coordination, this thesis proposes and validates an ultra-lightweight end-to-end reinforcement learning (RL) control scheme. The core contributions and engineering breakthroughs of this system are primarily manifested in the following three aspects:

- **Minimalist Architecture and Computational Liberation:** This research innovatively replaces traditional 2D image convolutional feature extraction with 1D grayscale projection features. Combined with an MLP network utilizing multi-modal concatenation (visual features and PSD distance), the single-frame end-to-end inference latency is strictly compressed to under 50 ms (20+ FPS). This architectural "burden reduction" thoroughly eliminates the control lag inherent to traditional CNNs, significantly unlocking the physical speed limits and agility of the actual robot.
- **Bridging the Reality Gap with Zero-Shot Transfer:** In response to the extreme and variable lighting disturbances on the real track, the system constructs a rigorous preprocessing pipeline integrating CLAHE, Gaussian blur, and Adaptive Thresholding. By cleaning high-frequency base noise at the physical level and coupling this with a visual Domain Randomization (noise injection) strategy during simulator training, the system flawlessly achieves robust deployment from simulation to the real-world environment without any real-world fine-tuning (Zero-Shot Transfer).
- **Physics-Compliant Cooperative Control:** Addressing the mechanical tendency of differential-drive chassis to slip, the designed kinematic difference penalty ($R_{kinematic}$) successfully guides the agent to autonomously master the physical balance between speed and steering. This thoroughly eradicates the fatal flaws of in-place sharp turning at startup and high-speed sideslipping. Simultaneously, in the physical dual-vehicle following experiments, the system demonstrated exceptional dynamic hazard avoidance and smooth speed regulation capabilities, achieving collision-free adaptive following in complex track and intersection environments.

5.2 Future Work

Although this system has achieved significant results in terms of lightweight design and robustness, based on the limitations discussed in Section 4.4, future research will primarily focus on the following two dimensions:

- **Hybrid-Dimensional Perception Architecture:** Addressing the loss of vertical semantic information caused by the 1D projection (e.g., the inability to recognize "STOP" signs), future work could consider designing a dual-branch architecture featuring "high-low frequency asynchronous fusion." This entails maintaining the 1D projection as the high-frequency backbone for high-speed tracking and obstacle avoidance, while running in parallel a very low-frequency (e.g., 5 FPS) lightweight 2D classification network specifically dedicated to capturing high-dimensional spatial semantics and traffic rule directives.

- **Visual Algorithmic Compensation for Perception Blind Spots:** To mitigate the issues of target loss and perception dependency bias associated with the single front-facing PSD sensor during sharp curves, future plans involve introducing an extremely lightweight Bounding Box Regression task directly from the existing visual stream. By endowing the network with visual tracking and auxiliary ranging capabilities for entities ahead, this approach fundamentally compensates for the geometric blind spots of a single-point ranging sensor at the algorithmic level. This will enhance the cross-validation capability of multi-modal features in highly dynamic scenarios, further guaranteeing the absolute safety boundaries during multi-agent cooperative interactions.

References

[1] T. Bräunl, "EyeBot Mobile Robots," 2026. [Online]. Available: https://roblab.org/eyebot/.

[2] T. Bräunl, "EyeSim Robot Simulation," 2026. [Online]. Available: https://roblab.org/eyesim/.

[3] V. Mnih, K. Kavukcuoglu, D. Silver and e. al., "Human-level control through deep reinforcement learning," *Nature,* pp. 529-533, 2015.

[4] A. Ng, D. Harada and S. J. Russell, "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping," *International Conference on Machine Learning (ICML),* 1999.

[5] S. Sun, J. Zheng, Z. Qiao, S. Liu, Z. Lin and T. Bräunl, "The Architecture of a Driverless Robot Car Based on EyeBot System," *Journal of Physics: Conference Series,* p. 012099, 2019.

[6] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. K. Yogamani and P. Pérez, "Deep Reinforcement Learning for Autonomous Driving: A Survey," *IEEE Transactions on Intelligent Transportation Systems,* pp. 4909-4926, 2020.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347,* 2017.

[8] S. Han, H. Mao and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *International Conference on Learning Representations (ICLR),* 2016.

[9] N. Jakobi, P. Husbands and I. Harvey, "Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics," *European Conference on Artificial Life (ECAL) / Lecture Notes in Artificial Intelligence,* pp. 704-720, 1995.

[10] K. Bousmalis, A. Irpan, P. Wohlhart and e. al., "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping," *2018 IEEE International Conference on Robotics and Automation (ICRA),* pp. 4243-4250, 2018.

[11] J. Tobin, R. Fong, A. Ray and e. al., "Domain randomization for transferring deep neural networks from simulation to the real world," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 23-30, 2017.

[12] A. Béres and B. Gyires-Tóth, "Enhancing Visual Domain Randomization with Real Images for Sim-to-Real Transfer," *Infocommunications Journal,* pp. 15-25, 2023.

[13] F. Sadeghi and S. Levine, "CAD2RL: Real Single-Image Flight without a Single Real Image," *arXiv preprint arXiv:1611.04201,* 2016.

[14] A. Y. Ng, D. Harada and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," *Proceedings of the Sixteenth International Conference on Machine Learning (ICML),* pp. 278-287, 1999.

[15] K. J. Zuiderveld, “Contrast Limited Adaptive Histogram Equalization,” *Graphics Gems IV,* 1994.