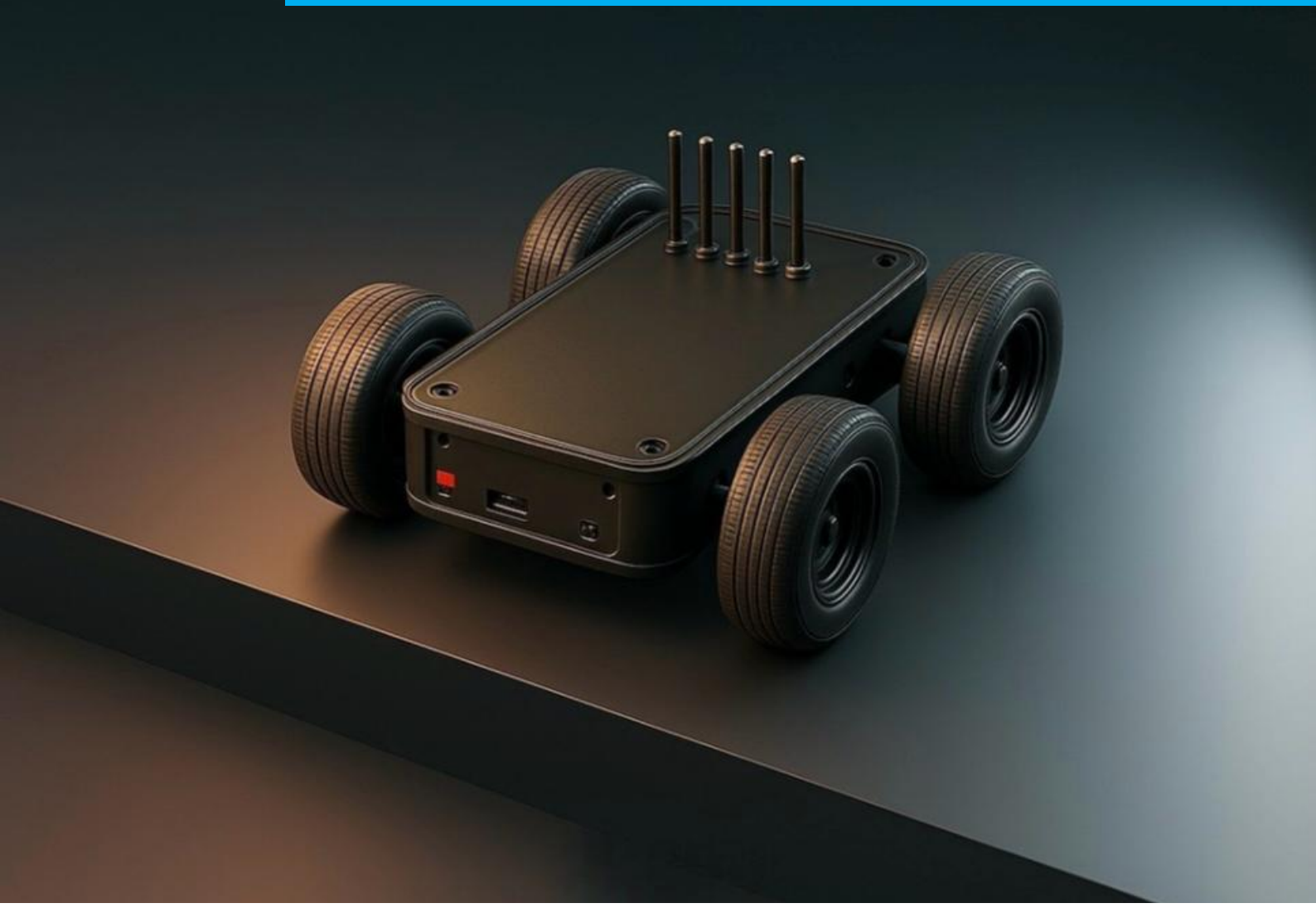
Department of Precision and Microsystems Engineering

Learning-Driven Torque Control for Skid-Steer Robots

M. Jansen

Report no : 2025.020
Coach : Dr. ir. J. Sijs
Professor : Dr. ir. H. Goosen, Dr. ir. E. J. Kober
Specialisation : Mechatronic System Design
Type of report : MSc Thesis
Date : July 1st, 2025



TU Delft Delft University of Technology

Challenge the future

Learning-Driven Torque Control for Skid-Steer Robots

Knowledge-Assisted Reinforcement Learning with Curriculum-Based System Identification for Trajectory Control

by

M. Jansen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on July 1st, 2025.

Student number:	4564030	
Project duration:	January 10, 2024 – July 1, 2025	
Thesis committee:	Dr. ir. J. Sijs,	Avular: Mobile Robotics
	Dr. ir. E. J. Kober,	TU Delft, supervisor
	Dr. ir. H. Goosen,	TU Delft, supervisor

This thesis is confidential and cannot be made public until July 1, 2025.

TU Delft

Preface

Reinforcement Learning (RL) offers a promising data-driven approach to decision-making in complex scenarios, but standard RL methods can struggle with the intricacies of terrain interaction and vehicle dynamics. Key challenges addressed include defining a proper problem definition, designing a curriculum learning strategy to gradually increase difficulty, and forms of imitation learning to overcome initial learning struggles. A low-fidelity controller is employed as an expert model to guide initial policy learning. The approach is validated in simulation experiments featuring progressively harder tasks to evaluate learning stability, performance, and generalization. We demonstrate that torque-based action formulations, enhanced by knowledge-assisted techniques, enable the agent to learn nuanced wheel-ground interaction, achieving accurate trajectory control.

Beyond the technical content, this work reflects a broader belief: that marrying human expertise with modern learning algorithms can accelerate progress in robotics. The knowledge-assisted frameworks developed here, (KAMMA, and their curriculum-augmented variants) embody this principle, blending structured guidance with autonomous learning. I hope that this thesis not only advances skid-steer control but also serves as a blueprint for future efforts to integrate domain knowledge and machine learning in other complex systems.

I would like to thank my supervisors for their guidance during this learning process and the many sparring rounds we shared. Furthermore, I dedicate this work to my family and friends, whose encouragement kept me focused through late nights and debugging marathons. May this thesis inspire others to tackle the intricate yet rewarding challenge of teaching machines to learn.

M. Jansen
Delft, June 2025

Abbreviations

RL Reinforcement Learning

MDP Markov Decision Process

DDPG Deep Deterministic Policy Gradient

KA-DDPG Knowledge-Assisted DDPG

KAMMA Knowledge-Assisted Mixed Mode Actioning

IL Imitation Learning

LSTM Long Short-Term Memory

PID Proportional–Integral–Derivative controller

CG Center of Gravity

GPU Graphics Processing Unit

TE Tracking Error

SM Smoothness

GV Growing Variance

BD Bimodal Drift

FIFO First In First Out

EMA Exponential Moving Average

List of Figures

List of Tables

List of Symbols

Symbol	Meaning
s	MDP state
a	MDP action
r	Reward
s'	Next state
G_t	Discounted return
γ	Discount factor
π	Policy
$\mu(s \mid \theta_\mu)$	Actor (deterministic policy)
μ_f	Friction coefficient
$Q(s, a \mid \theta_Q)$	Critic / Q-function
θ_μ, θ_Q	Network parameters
τ	Soft-update coefficient (target nets)
$T_{\min}$, $T_{\max}$	Torque bounds
m	Vehicle mass
B	Track width
L_f, L_r	Front/rear axle distance to CG
R	Wheel radius
J	Inertia about CG
j	Rotation moment of inetia for the wheels
e_v, e_ω	Velocity and yaw-rate error
e'_v, e'_ω	Time derivatives of those errors
α_t, β_t	Blending (action and reward) weights
π_{expert}	Expert-action probability

Table 1: Key symbols from the thesis and their usage consistency.

Contents

1 Introduction

1.1. Motivation and Problem Statement

Robotic reinforcement learning (RL) has seen promising successes in recent years on complex locomotion and navigation tasks. For example, Hoeller et al. (2023) demonstrated that RL can enable a quadrupedal robot (ANYmal) to perform parkour-like maneuvers, such as leaping over gaps and climbing obstacles. These achievements underscore how training in simulation and large-scale learning can push robot capabilities. Richard Sutton's "bitter lesson" (Sutton, 2019) further argues that methods scaling with compute and data tend to outperform those reliant on expert manual engineering. In other words, general learning approaches eventually win out over task-specific methods as more experience is gathered.

Applying this insight to wheeled mobile robots, we aim to leverage data-driven learning for dynamically complex control problems. However, solely relying on data and RL algorithms to solve these challenges is often impractical for robotics precisely due to a shortage in data and compute. Training an agent from scratch to perform precise torque control can require prohibitively many trials and may expose the robot to unsafe behaviors early on. A pragmatic middle ground is to incorporate expert knowledge in the learning process as guidance rather than as a rigid control structure. By using structured training curricula and expert demonstrations as flexible guides, the learning process can be accelerated while still allowing the agent to ultimately discover an optimized policy on its own.

Figure 1.1: Avulars' Skid-steering Origin One vehicle

In this thesis, we focus on skid-steer mobile robots, differential-drive vehicles such as Avulars' origin one platform shown in Figure 1.1 (the focus platform of this research). We will focus on the task of learning low-level torque control for accurate trajectory following. Skid-steer robots, often employed for rough-terrain navigation and maneuverability in constrained spaces, represent a class of non-holonomic vehicles whose dynamic behavior poses significant challenges for RL-based methods.

A critical problem encountered when employing RL for skid-steer robots is the phenomenon of wheel slip. Skid-steer robots rely on slip to turn, and reaching different velocities means they experience many different traction regimes while driving. Classic robot controllers often assume constant idealized kinematics with negligible slip. However, slope angles and friction coefficients make those assumptions nonviable when looking for deployment in the real world. With good traction the relationship between the applied torque and the resulting movement is more predictable. However, as friction decreases, wheel slip becomes more pronounced and the system's dynamics become more non-linear and uncertain. These challenges transfer to RL training where slip creates inconsistent data that undermines reliable policy training. This inconsistency severely limits the effectiveness of conventional RL methods, as policies trained on noisy or unreliable data struggle to converge to stable solutions.

Another crucial challenge is the dynamic coupling between the wheels, a characteristic of skid-steer vehicles. Due to this coupling, each wheel's action directly impacts the other wheels, significantly constraining the feasible control actions. This interconnectedness complicates action selection, reducing the action space's effective dimensionality and making conventional RL approaches highly data-inefficient.

Skid-steer robots are non-holonomic systems, meaning their orientation and forward motion are intrinsically linked and thus path- or history-dependent. This coupling intensifies the classic "credit assignment problem" in RL: the difficulty of correctly attributing successful or unsuccessful outcomes to specific actions. When you task an agent with a long-horizon goal—say, driving ten meters to a target or climbing over a step—it may receive reward only upon completion, leaving it unclear which individual actions actually mattered. The same ambiguity arises when we reward the robot for reaching a particular velocity. That velocity results from many preceding control commands, yet most RL algorithms update their models using only the immediate state and action. This is typical under a Markov Decision Process (MDP) assumption which states that the next best action depends solely on the current state. Because these updates rely on very local, step-by-step differences, the agent has no built-in way to integrate evidence over time. This makes it hard to understand the patterns of dynamic coupling and non-holonomic constraints. One could introduce memory mechanisms—like Long Short-Term Memory (LSTM) layers—to carry information across steps. But most RL algorithms lean solely on generalization—the ability to take patterns learned in one situation and apply them to new but similar situations. This generalization mechanism is capable of bridging the gap between a vague reward signal and action decisions. Unfortunately data consistency is a key-problem for this understanding, and skid-steer vehicles also suffer from nonlinear and uncertain traction conditions, which make it harder to form reliable generalizations.

In response to these challenges, this thesis adopts torque-based actions as a key design choice. Prior work on skid-steer RL found that purely velocity-driven policies often demand wheel motions that the physical system cannot realize, leading to excessive slip or instability (Dai et al., 2022). In contrast, torque-based actions require the agent to learn the dynamics (friction, inertia) but ensure physical consistency by design. The agent cannot "cheat" physics, it must discover how to generate appropriate forces. This expressiveness comes at the cost of a harder exploration and generalization problem. Although a detailed comparative justification between torque and velocity actions will be discussed in Section 2, torque selection inherently tries to mitigates slip-related problems.

To make RL feasible for complex physical systems, researchers have increasingly integrated domain-specific expert knowledge into the training process. Techniques such as curriculum learning, system identification, and imitation learning (IL) have proven effective. Rather than fully specifying a model's dynamics or parameters upfront, these methods progressively guide the learning process, leveraging prior knowledge to enhance efficiency and safety. Such approaches strive to achieve robust, interpretable policies without necessitating massive "foundation" models trained on enormous, general-purpose datasets (Hu et al., 2023).

This thesis addresses the above complexities by combining scalable RL algorithms with strategically

injected domain expertise. The core idea is to guide the agent's exploration with two key ingredients: (1) a curriculum that gradually increases task difficulty, and (2) expert demonstrations that provide an initial policy prior or assist the agent during early training. Our approach is to let the agent start learning under easier conditions and under the tutelage of a low-fidelity expert controller. Then progressively hand over control to the agent and increase the task demands. In doing so, we harness the best of both worlds: the rapid initial progress from expert guidance and the asymptotic performance of autonomous RL. By the end of training, the agent policy should be entirely self-driven (no expert input), capable of robust trajectory tracking under full difficulty conditions.

1.2. Research Question

Based on the above motivation, the central research question of this thesis is:

"How can expert knowledge be systematically integrated—via RL task design, curricula, and demonstration priors—to enable robust and data-efficient torque control learning in nonholonomic mobile robots?"

This question is broken into three specific sub-questions:

1. **RL Problem Design:** How do different problem formulations affect learning? In particular, how does the action representation (torque-based vs. velocity-based, individual control vs differential control) influence learning efficiency, terrain interaction, and overall control performance? What role do other design aspects (observation inputs, reward shaping, control frequency) play?
2. **Curriculum-Based Training:** Can gradually increasing the task complexity – from simpler maneuvers to the full difficult trajectory-following task – extend the low-fidelity control envelope? What is the impact of curriculum scheduling on learning speed and final performance?
3. **Expert-Assisted Learning:**How effective is leveraging an expert policy to assist the RL agent? Specifically, can using a low-fidelity controller to "warm-start" or guide the agent, result in faster convergence and better final performance compared to learning entirely from scratch, or only from imitation?

Underpinning these questions is the hypothesis that combining curriculum learning with expert-assisted RL will significantly improve learning outcomes. We hypothesize that an agent trained with both staged tasks and expert guidance will learn faster and achieve more reliable trajectory tracking than an agent using either technique alone (or neither). By systematically investigating each component (problem design, curricula, and demonstrations), we aim to validate this hypothesis and pinpoint the contributions of each.

1.3. Contributions and Thesis Scope

The contributions of this research are summarized as follows:

1. **Comparative Analysis of Action Spaces:** We offer an in-depth empirical comparison of several action-space formulations for skid-steer control. We briefly examine naive velocity-based policies, which tend to command unrealizable wheel speeds and suffer from excessive slip and poor traction (Dai et al., 2022). We go on to use torque-based commands that enforce physical consistency by forcing the agent to learn true friction and inertia dynamics. This yields more accurate control at the cost of a tougher exploration problem. Then we examine the problem of dynamic coupling with differently dimensioned torque-based action spaces. We consider individual wheel control, a differential-drive approximation that groups wheels into left-and-right actuators, and a single global torque profile: this simplification reduces the action dimension and eases training, but restricts the robot to standard curvature and pivot motions. Altogether, our analysis shows that richer action spaces boost control fidelity while amplifying exploration difficulty, whereas lower-dimensional schemes streamline learning by trading off expressive capacity (Eßer et al., 2024).
2. **Knowledge-Assisted Mixed Mode Actioning (KAMMA):** We introduce a novel knowledge-assisted RL strategy called KAMMA. Building on the baseline Knowledge-Assisted DDPG framework of Dai et al. (2022), KAMMA replaces continuous action blending with a probabilistic switch-

ing mechanism reminiscent of ϵ-greedy methods. At each time step, either the expert's action or the agent's action is executed in its entirety, with a decaying probability of choosing the expert. This mixed-mode action selection preserves training stability (thanks to expert oversight early on) while avoiding the interference and torque saturation issues observed with continuous action blending. KAMMA simplifies the learning signal for the agent and leads to smoother policy development as one of our key innovations.

3. **Curriculum Design for IL Training:** We develop multiple curriculum learning schemes tailored to the skid-steer trajectory tracking task. In each curriculum, the agent begins with easier scenarios – for instance, following slow-speed trajectories – and progressively moves to harder conditions (higher speeds). We explore two variants of curricula to identify what scheduling works best. These curricula are informed by existing frameworks in curriculum RL (e.g., the incremental approach of Margolis et al. (2022)) and are customized to our torque-control domain. Through a systematic evaluation, we demonstrate that curriculum learning can improve training stability, and final tracking accuracy.

4. **Integrated IL+RL Training (KAMMA + Curriculum):** We present a unified training approach that combines expert demonstrations with curriculum learning. To our knowledge, this work is the first to systematically evaluate the integration of multiple curricula in an expert-guided RL setting (addressing a gap in skid-steer literature identified in Section 3.4). By incorporating expert guidance (via KAMMA) across a staged curriculum, we enable the agent to master fundamental skills under expert supervision and then exceed the expert's capabilities as the difficulty increases. Our results show that this integrated approach yields a larger control envelope of more robust final performance than using either expert assistance or curricula alone. In essence, we extend the ideas of Margolis et al. (2022) (who compared curricula) by adding the dimension of expert assistance, and extend Dai et al. (2022) (who used knowledge assistance) by applying it across a curriculum of tasks.

5. **Comprehensive Experimental Validation:** We implement and rigorously evaluate all proposed methods in a high-fidelity physics simulation environment, NVIDIA Isaac Sim (Salimpour et al., 2025). The experimentation includes ablation studies and baseline comparisons. We measure quantitative metrics such as trajectory tracking error and policy smoothness. The evaluation covers multiple action space settings and training strategies to provide robust evidence for each contribution. All comparisons are run across multiple random seeds to ensure statistical significance. The end result is a set of robust baselines and insights for skid-steer robotic learning under realistic conditions, which can inform and accelerate future research in this domain.

Overall, the scope of this thesis spans from foundational design decisions (state and action representations, reward shaping) to high-level training strategies (curriculum and demonstration integration), unified under the goal of reliable skid-steer control. We emphasize the novel KAMMA mechanism and its extension with curriculum learning as the primary contributions, with comparative analyzes and experimental insights supporting these main innovations.

1.4. Document Road-map

The remainder of this thesis is organized as follows:

- **Chapter 2 – General Background:** Introduces the necessary background on RL for continuous control, imitation learning, and curriculum learning. These sections provide theoretical context for our approach, clarifying concepts like actor–critic RL, expert demonstration, and staged training in robotics.

- **Chapter 3 – Related Work:** Reviews relevant literature in four areas: (3.1) RL applied to nonholonomic vehicle control, (3.2) the use of curriculum learning in robotic training, (3.3) demonstration-enhanced and knowledge-assisted RL methods, and (3.4) Combining IL and curriculum learning. We highlight key findings from prior studies and identify a gap – namely, the lack of studies combining expert guidance with curricula in one framework. Section 3.5 synthesizes the related works and positions our research contributions in relation to this gap.

- **Chapter 4 – Methodology: From KA-DDPG to KAMMA + Curriculum:** Details our approach.

We begin in Section 4.1 with the problem formulation, defining the state observation, and action spaces, and the reward function for our skid-steer RL problem. Section 4.2 describes the baseline knowledge-assisted RL implementation (KA-DDPG) and discusses its limitations. Building on this, Section 4.3 introduces the KAMMA strategy as an improved action selection mechanism. Section 4.4 then integrates KAMMA with curriculum learning, outlining the design of the training phases and performance-based progression criteria.

- **Chapter 5 – Experiments and Results:** Presents the experimental setup and empirical results. Section 5.1 covers the used metrics. Section 5.2 covers the baseline experiments analyzing different action space configurations (4D, 2D, and 1D torque control). Section 5.3 evaluates our KAMMA method against the baseline KA-DDPG (and other ablations) in the full 4D torque control task. And Section 5.4 reports the performance of the combined KAMMA + Curriculum approach, including comparisons between two curriculum variants. Throughout this chapter we report tracking error, and other metrics, with commentary on the observed behaviors.
- **Chapter 6 – Discussion:** This chapter synthesizes the experimental findings and critically reflects on the design decisions, empirical trade-offs, and open challenges encountered. Section 6.1 consolidates the key empirical insights across all experimental configurations, highlighting how action dimensionality, curriculum scheduling, and expert blending impacted performance. Section 6.2 examines unexpected observations and methodological implications—such as the influence of probabilistic versus blended action selection and the role of curriculum in mitigating catastrophic forgetting. Section 6.3 outlines core limitations of the current approach, including data bias, exploration constraints, and transferability concerns, alongside possible mitigation strategies. Section 6.4 extends the discussion to broader research implications, identifying future avenues in curriculum design, replay strategy, and model-based extensions. Finally, Section 6.5 summarizes the high-level conclusions drawn from the study.

By following this document road-map, the reader is first grounded in background and literature, then guided through our methodological innovations, and finally presented with a thorough evaluation and reflection on the outcomes.

2

General Background

2.1. Reinforcement Learning for Continuous Control

Torque-based robotic control presents a unique challenge for reinforcement learning. Unlike position or velocity control, torque commands affect system dynamics directly through acceleration, requiring agents to model latent physical relationships such as friction, inertia, and slip. This makes the credit assignment problem particularly difficult, especially in long-horizon tasks.

We frame this control problem as a Markov Decision Process (MDP), which assumes that the system's dynamics depend only on the current state and action. While this simplification enables theoretical guarantees and algorithmic tractability, it clashes with the physical reality of many non-holonomic systems. In such cases, actions have delayed and entangled effects—a fact that exacerbates learning instability when regular RL methods are used.

To handle continuous control, we model the response to the environment through function approximation with neural networks and rely on actor–critic architectures to manage high-dimensional state and action spaces. All these components and their relationship to the temporal credit assignment problem will be explained in detail in the following subsections.

2.1.1. Markov Decision Processes

RL problems are typically formalized as MDPs (Sutton & Barto, 2018). An MDP is defined by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, and a transition model $p(s', r \mid s, a)$ giving the probability of next state s' and reward r given current state s and action a. At each timestep t, an agent observes state s_t, takes action a_t according to its policy, and receives a reward r_t from the interpreter, like shown in Figure 2.1. The goal is to find a policy (a mapping from states to actions) that maximizes the cumulative reward over time. Typically this is done by maximizing the expected return, defined in equation 2.1 below. The Markov property ensures that $p(s', r \mid s, a)$ depends only on the current state and action, not the full history.

2.1.2. Return and the Bellman Equation

The return G_t at time t is the cumulative (discounted) future reward from that point:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2.1}$$

Where $0 \leq \gamma < 1$ is a discount factor that ensures convergence (Sutton & Barto, 2018). The value function of a policy π is the expected return: $v^\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s]$. Similarly, the action-value function (or Q-function) is $q^\pi(s, a) = \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a]$. These functions satisfy recursive relationships known as the Bellman equations. This enables recursive updates. In particular, under policy π the

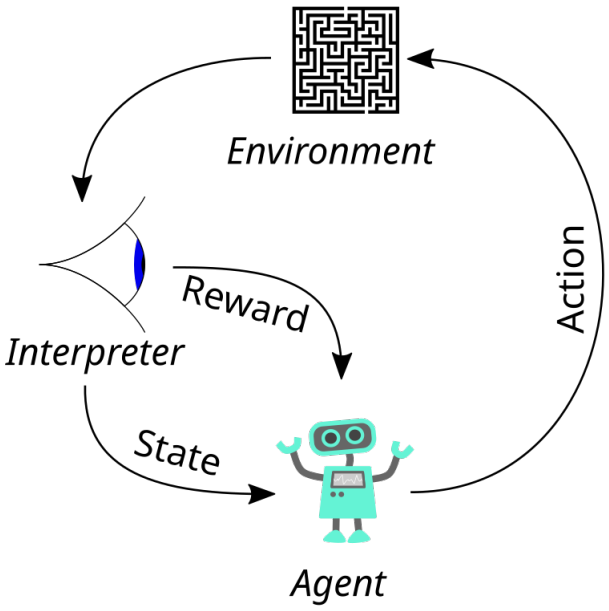


Figure 2.1: MDP components (Scomparin et al., 2024)

state-value and action-value functions obey:

$$v^{\pi}(s) = \mathbb{E}[r + \gamma v^{\pi}(s') \mid s], \quad q^{\pi}(s,a) = \mathbb{E}[r + \gamma q^{\pi}(s',a') \mid s,a], \tag{2.2}$$

where the expectation is over the transition s' and the next action $a' \sim \pi(\cdot \mid s')$.

2.1.3. Function Approximation and Actor–Critic Architectures

For complex or continuous state/action spaces, it is infeasible to represent value functions or policies in a lookup table. Instead, function approximators (typically neural networks) are used to estimate $v(s)$, $q(s,a)$, or the policy $\pi(a \mid s)$. However, large nonlinear approximators can make learning unstable (Lillicrap et al., 2015). To cope with this, modern methods often use actor–critic architectures. An actor–critic method maintains two models: an actor, which parametrizes the policy $\mu(s \mid \theta^{\mu})$ (deterministically mapping states to actions), and a critic, which estimates a value function $Q(s,a \mid \theta^{Q})$. The critic evaluates how good the actor's actions are, and the actor is updated to improve its performance according to the critic's feedback.

2.2. Deep Deterministic Policy Gradient (DDPG)

We make use of DDPG ((Lillicrap et al., 2015)) as this is a model-free, off-policy reinforcement learning algorithm well-suited for continuous action spaces like torque control. It leverages an actor–critic architecture and experience replay to learn deterministic policies from previously collected data (See sction 2.2.2 for details). This architecture is advantageous in simulation-heavy tasks where real-time feedback is costly or unsafe. It extends the Deterministic Policy Gradient (DPG) approach (Silver et al., 2015) by using deep networks. In DDPG, both the actor $\mu(s \mid \theta^{\mu})$ and the critic $Q(s,a \mid \theta^{Q})$ are represented by neural networks. Learning proceeds by repeatedly sampling transitions (s,a,r,s') and updating both networks through the Bellman equations.

2.2.1. Off-Policy

DDPG is called an off-policy algorithm because its critic network directly takes a state–action pair (s,a) as input and learns their value independently of the current transition. The input-output structure of the Actor and Critic networks are shown in Figure 2.3. You can see how the actions are used as input for the Critic while this action does not have to come directly from the Actor.

Because the critic does not actually require a to come from the current actor π_{ϕ}, we can inject arbitrary actions into the buffer—whether from random exploration, older policies, or expert demonstrations—

and immediately learn their values. This flexibility allows us to influence the data distribution by adding new (s, a) examples, which accelerates learning and can help steer the agent away from unsafe or uninformative regions. This is crucial for the problem of skid-steer where the usable part of the action space is very narrow and difficult to find as briefly summed up in the Introduduction.

In contrast, on-policy methods restrict updates to actions sampled from the current policy, tightly coupling data collection with policy optimization. DDPG's off-policy design decouples these steps, enabling far greater flexibility and data efficiency when training torque-based controllers in complex, expensive simulators.

2.2.2. DDPG Framework

Figure 2.2 shows the basic framework of a DDPG algorithm with its components described below:

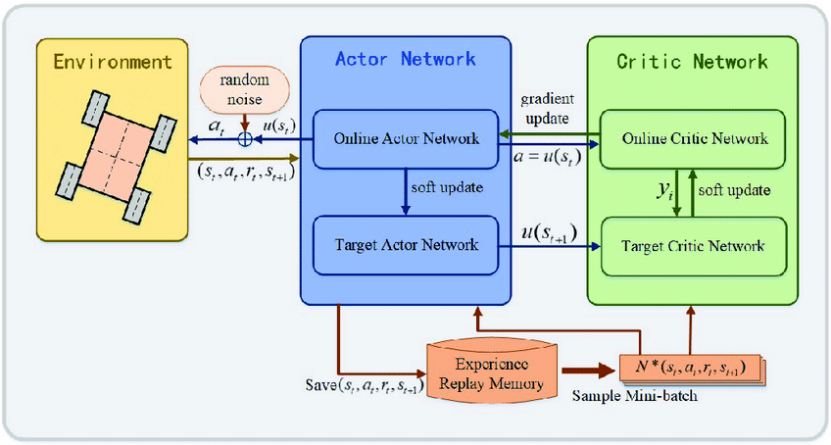


Figure 2.2: Framework of DDPG algorithm (Dai et al., 2022)

Experience Replay Memory DDPG stores experienced transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ in a replay buffer. During learning, minibatches of transitions are sampled uniformly from this buffer to update the networks. This breaks the temporal correlations between consecutive samples and improves data efficiency by reusing past experiences. DDPG's off-policy replay buffer treats all past transitions as equally valid training samples, implicitly assuming that the state-action distributions under older policies remain representative of the current policy's dynamics. Yet as weights evolve, the learned policy explores new regions of state–action space that may never have been visited under earlier behaviors. Conversely, data from very early training—when exploration noise dominates—may become irrelevant or even misleading. This distribution mismatch can bias the critic toward outdated or off-distribution values and misguide the actor's updates. In extreme cases, the agent may unlearn effective behaviors because the replay buffer is flooded with low-quality transitions from early exploratory noise phases. Managing the replay buffer can be an important aspect to off-policy algorithms.

Target Networks Two additional "target" networks μ' and Q' are maintained as delayed copies of the actor and critic. The target networks have parameters $\theta^{\mu'}$ and $\theta^{Q'}$ that are updated slowly for stability reasons towards the main network parameters:

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}, \quad \theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}. \tag{2.3}$$

Target values for the Bellman update are computed as $y = r + \gamma Q'(s', \mu'(s'))$. This is the bootstrap estimate of the expected return. Bootstrapping refers to the practice of estimating the value of a state (or state-action pair) using another estimate, in this case using the target network to determine the Q-value. Bootstrapping can introduce instability because the target value depends on another function approximator — and both the network doing the learning (main critic) and the one providing the target (target critic) are updated frequently, leading to moving targets. Hence the introduction of a slowly updated target.

Bellman Error Loss (Critic Update) The critic network is trained by minimizing the mean-squared Bellman error:

$$\mathcal{L}(\theta^Q) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q(s,a \mid \theta^Q) - y\right)^2\right], \quad y = r + \gamma Q(s', \mu(s') \mid \theta^Q). \tag{2.4}$$

In practice, y is computed using the target networks, i.e., $y = r + \gamma Q'(s', \mu'(s'))$. This is where the stability is strengthened with target networks in DDPG, since otherwise the Loss would be chasing a moving target.

Deterministic Policy Gradient (Actor Update) Following the critic update via the bootstrapped target described earlier, the actor network is optimized to improve the expected return by directly leveraging the critic's estimates. Specifically, the Deterministic Policy Gradient (DPG) Theorem is applied, which allows the agent to compute the gradient of the expected return with respect to the actor parameters θ^μ. The objective is to adjust the policy such that it outputs actions which the critic estimates as having high value. Mathematically, the actor update seeks to maximize the Q-value of the state-action pair under the current policy. To compute the gradient for this objective, the chain rule is applied via:

$$\nabla_{\theta^\mu} J(\theta^\mu) = \mathbb{E}_{s\sim\mathcal{D}}\Big[\nabla_a Q\big(s, a \mid \theta^Q\big)\big|_{a=\mu(s|\theta^\mu)} \nabla_{\theta^\mu} \mu(s \mid \theta^\mu)\Big]. \tag{2.5}$$

Here, the critic provides the action-value gradient, indicating in which direction the action should move to increase expected return, and the actor adjusts its parameters to shift its output policy in that direction. This effectively aligns the actor with the critic's assessment of optimal behavior, treating the critic as a differentiable objective function over actions.

This update process is depicted in 2.3 and relies on the assumption that the critic is reasonably accurate which, as discussed in the previous section, is made more stable through the use of target networks and bootstrapped targets. By ensuring that the critic's predictions are smooth and slowly varying over time (thanks to the soft update mechanism of the target networks), the actor can learn from consistent gradients, reducing the risk of policy oscillation or divergence during training.

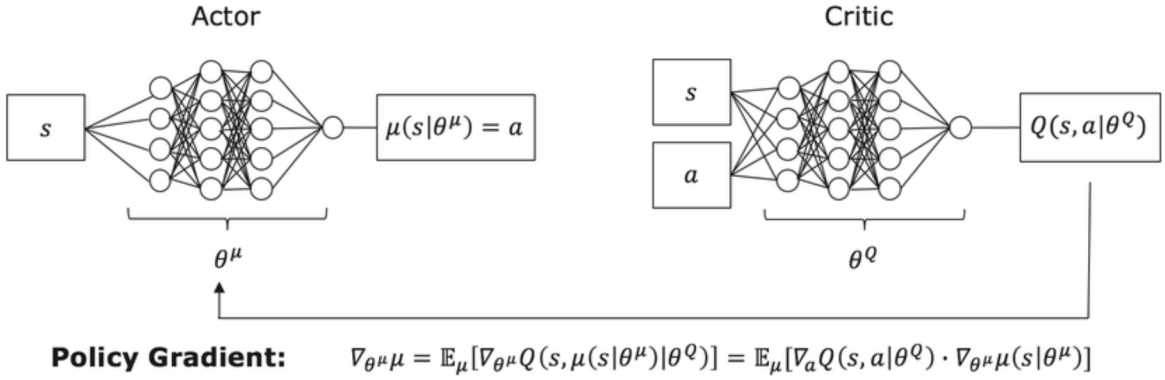


Figure 2.3: Basic structure of the DDPG actor-critic agent (Liessner et al., 2018)

Exploration Noise Since $\mu(s \mid \theta^\mu)$ is deterministic, DDPG injects exploration noise during data collection, often by adding temporally correlated Ornstein–Uhlenbeck noise or Gaussian noise:

$$a_t = \mu(s_t) + \mathcal{N}_t \tag{2.6}$$

Ergodicity in RL is about ensuring that your exploration policy can, over time, sample the full breadth of behaviors you care about. In continuous or non-holonomic settings, true ergodicity is impossible—but being aware of which regions your policy never visits helps you diagnose blind spots and design

exploration or replay strategies that mitigate them. One can only hope to densely sample small neighborhoods around frequently visited states because the Ornstein–Uhlenbeck noise only injects local perturbations around the deterministic policy. If crucial regions, such as high-slip or extreme steering configurations, are never sampled, the critic will have no basis to evaluate them accurately, and the actor can never learn the correct corrective torques for those regimes.

2.2.3. Considerations

These mechanisms collectively allow DDPG to learn continuous-domain policies in high-dimensional problems. The replay buffer and target networks stabilize the learning process, the Bellman loss trains the critic, and the deterministic policy gradient updates the actor. Despite these tools, DDPG remains vulnerable to sample inefficiency in long-horizon tasks with sparse rewards. When slip occurs at high torque, reward spikes, or noise, can mislead the critic for dozens of timesteps. Sparse rewards (e.g. reaching a target position) delay feedback, making it unclear which early actions contributed to the failure. Dense rewards (e.g. penalizing velocity error or heading deviation at each timestep) alleviate this by offering more frequent training signals. However, they can also bias the agent toward locally optimal behaviors if not carefully designed. A good solutions warrants both dense and sparse rewards.

RL on Torque-Controlled Systems Applying RL to torque control tasks introduces challenges specific to the physics of the system. Torque corresponds to acceleration, meaning that the agent must learn the cumulative effect of torque over time to achieve velocity or positional objectives. However, standard RL policies are stateless with respect to time: they produce actions based only on the current observation.

For example, a policy $\pi_\phi(s)$ that outputs a torque command at time t does so, based only on s_t. This state can include state observations from a number of steps back, but these additions are useless without a precise and stable integration mechanism, like available with ordinairy differential equations. Then the past torques could be used to form velocity or position generalizations. But because accelerations are some of the noisiest state signals in this setup, temporal credit assignment becomes nontrivial: if success is measured at the positional level, the agent must infer which torque sequences were responsible.

This is exacerbated by the nonholonomic nature and dynamic coupling of skid-steer robots: their constraints couple the rotational and translational dynamics in a way that makes certain paths infeasible or state dependent. Moreover, slip between wheels and terrain adds noise to state transitions, complicating the learning process by introducing stochasticity in $P(s'|s,a)$. Separately learning a dynamics model $P(s'|s,a)$ can reduce sample stochasticity, but incurs bias if the model is imperfect, potentially affecting long-term planning and credit assignment.

2.2.4. Conclusion

RL-particularly off-policy actor-critic methods like DDPG-offers a promising route to train torque-level controllers for skid-steer robots. However, the memoryless nature of MDP-based RL, combined with the physics of non-holonomic systems, makes temporal credit assignment and reward design essential challenges. Understanding these mechanisms lays the foundation for the subsequent contributions of this thesis, where expert-informed guidance and curriculum shaping will be introduced to address these structural limitations.

2.3. Imitation Learning and Demonstration Guidance

Imitation Learning (IL) is a paradigm where an agent learns to mimic expert behavior from demonstrations, rather than learning purely from its own trial-and-error. In robotics, IL often takes the form of behavior cloning (BC), where a dataset of state-action examples from an expert (human operator or a heuristic controller) is used to supervise the training of a policy network. In its pure form, behavior cloning minimizes the mean squared error (MSE) between the expert's and learner's actions:

$$\mathcal{L}_{\text{BC}} = \mathbb{E}_{s\sim\mathcal{D}}\left[\|\pi(s) - \pi_E(s)\|^2\right], \tag{2.7}$$

where is a dataset of state-action pairs collected from the expert. The appeal of imitation is that it can provide an excellent initial policy offline, jump-starting the agent near a good solution and avoiding the

random exploration that a fresh RL agent might exhibit.

In RL contexts, this concept can be translated into a reward-shaping mechanism. Rather than explicitly minimizing a loss in a supervised setting, the agent receives high reward when its action is close to the expert's action. For instance, a reward function for imitation can be defined as:

$$R(s, a) = -\|a - \pi_E(s)\|_2, \tag{2.8}$$

where a smaller distance (higher similarity to the expert action) yields a higher reward. This formulation turns the imitation objective into a RL signal, enabling its integration with standard RL algorithms such as DDPG.

However, pure IL has limitations. A policy trained only by cloning an expert will at best replicate the expert's performance; it may not improve beyond the expert and could be brittle if it encounters states not covered in the demonstrations. If the agent encounters novel states it can drift into compounding errors (the covariate shift problem), since it has no informed feedback for those unseen situations (Cimurs & Merchán-Cruz, 2022). In our context, we have access to a low-fidelity expert controller for skid-steer trajectory following (for example, a simplified P controller that tries to track velocities). We expect this expert to be suboptimal – perhaps it can handle only low-slip conditions well, or it follows trajectories somewhat imprecisely. If we rely solely on imitation, the agent would inherit these deficiencies and plateau at the expert's level. Sutton (2019) emphasizes the importance of allowing the agent to continue learning from real experience to discover strategies that even a human or heuristic might not have considered. In practice, this means using imitation to initialize or guide the agent, but not letting it stunt the agent's growth.

Incorporating demonstrations into RL can be done in several ways. One way is pre-training: first trains the policy network on demonstrations (supervised learning) to obtain a reasonable policy, then switch to RL to further improve it. Another way, which we adopt, is off-policy guidance: during RL training, occasionally leverage the expert's action or feedback to keep the agent on track. This could be as simple as starting each training episode with the expert policy for a few steps (to put the agent in a good state), or more continuous forms of guidance like blending expert actions with the agent's actions. The approach by Han et al. (2025) is illustrative: they pre-trained a policy on simulation demonstrations and later fine-tuned it on the real robot with a few real-world demonstrations as anchors. This significantly improved learning efficiency and success rate compared to pure RL.

When using imitation in training, an important consideration is when to rely on the expert and when to let the agent try its own actions. If the expert's influence is too strong for too long, the agent may never discover better strategies than the expert's. On the other hand, if the expert is removed too early, the agent can flounder and converge to a poor policy. A common solution is to schedule the expert's involvement to decay over time – heavily guide the agent initially, and gradually reduce assistance as the agent gains competence. This ensures a smooth transition from apprenticeship to autonomous learning. Another practical consideration is the cost of obtaining demonstrations. In some cases, demonstration data might come from a human teleoperating the robot or from running a computationally expensive optimal controller. These can be costly, so we want to use the expert data efficiently. Ideally, a small number of demonstrations can yield a large benefit in jump-starting learning.

To summarize, IL provides a powerful kickstart for RL in robotics by addressing the exploration cold-start problem – the agent is less likely to take utterly random (and unsafe) actions if it has an expert to imitate initially. It also offers some safety guardrails: the expert can prevent catastrophic failures early in training. However, imitation should be combined with RL so that the agent can continue improving. Our approach will integrate IL in a way that the agent is warm-started but not constrained by the expert in the long run. We will monitor and control the agent's reliance on the expert to ensure it eventually becomes fully independent and surpasses the expert's performance.

2.4. Curriculum Learning Concepts

Curriculum learning involves training an agent on a series of tasks that increase in difficulty, rather than throwing the hardest task at the agent from the beginning. The concept is inspired by human education – we learn simpler skills first and build up to more complex ones. In RL, curriculum learning has proven effective in domains where direct learning on the hardest task fails or is very slow. By solving easier

sub-tasks first, the agent acquires foundational skills and confidence that can later help in mastering the final task.

Formally, Narvekar et al. (2020) provide a comprehensive framework and survey for curriculum learning in RL. They categorize curricula into task-based curricula, where the agent is trained on a sequence of entirely different tasks (each task building on skills from prior ones), and domain-based curricula, where the task remains the same but some environment parameters are varied to adjust difficulty. In both cases, the idea is to present the agent with learning experiences that are appropriate to its current skill level. Narvekar et al. (2020) also distinguish between manually designed curricula and automated curriculum generation algorithms. In automated curricula, the sequence of tasks is generated by an algorithm (often by formulating a meta-learning or two-player formulation where one agent proposes tasks and the other learns), whereas in manual curricula, the researcher defines the progression based on domain knowledge.

In curriculum design, an important aspect is defining stage transition criteria. Rather than switching phases after a fixed number of episodes, it is often effective to use performance thresholds. For example, we might say the agent can advance to the next stage after it achieves, say, 5 consecutive successful episodes at the current stage (where success could mean tracking error below a certain threshold). This ensures the agent does not move on without sufficiently learning the current task.

This principle has been validated in prior work. Rudin et al. (2021), for example, demonstrated a powerful adaptive curriculum where a quadruped was trained to walk on challenging terrains ”in minutes” by automatically increasing terrain difficulty whenever the agent achieved a certain proficiency. This ensured the agent was always training at the edge of its capability, maximizing learning efficiency.

Building on these insights, the curriculum in this thesis is not based on a two-phase setup but instead follows a continuously increasing difficulty scheme. Rather than distinguishing tasks as simply “easy” or “hard” based on speed, we define task difficulty in terms of proximity to the operational envelope of a low-fidelity expert controller. Tasks close to the expert controller’s capabilities are considered easier, while those requiring actions at or beyond the limits of the expert’s performance are considered more difficult. This framing is particularly appropriate in torque-based control, where physically plausible action sequences are critical and dangerous behaviors must be avoided.

Studies show that curricula not only improve training stability and convergence speed, but also generalize better to unseen states—an area where pure IL can be brittle. By introducing structure into the agent’s exposure to complex tasks, curriculum learning complements the role of expert demonstrations.

In summary, curriculum learning is a powerful technique to shape the learning process for difficult tasks. It addresses the exploration problem by initially limiting the scope (the agent is less likely to encounter states it can’t handle early on) and provides a form of shaping for the state-space and reward landscape. In this thesis, curriculum learning will be used in conjunction with expert guidance as complementary tools: the curriculum tackles the task complexity aspect, while expert demonstrations tackle the exploration initialization aspect. By the end of training, we aim for an agent that has learned to handle the full task complexity entirely on its own, thanks to having a proper stepping stone path to get there.

2.4.1. Curriculum-Driven System Identification

Beyond easing exploration, structured curricula can also serve a deeper role: enabling the agent to uncover patterns in the system’s basic dynamics. In our setup, the early curriculum stages — characterized by low speeds and minimal accelerations — expose the agent to clear, simplified relationships between torque inputs and resulting motion. Rather than estimating friction coefficients or inertia matrices explicitly, our curriculum uses staged exposure to reveal these dynamics to the agent:

- **Stage 1: Low-speed.** Torque commands are restricted to a narrow band around the expert’s nominal working point. Here, the agent learns the friction threshold required to initiate motion—akin to measuring static friction in classical system ID.
- **Stage 2: Mid-range velocities.** We gradually move away from the controllers working point and start experiencing more uncontrolled slip. The agent learns how torque maps to slip and

traction loss. This mirrors how a system-ID experiment would sweep an input to measure slip characteristics.

- **Stage 3: Edge-of-expert domain.** Commands approach the expert's operational limits, forcing the agent to infer complex, speed-dependent friction, and wheel-coupling effects. At this point, the curriculum has likely helped the agent approximate key dynamics (friction, mass) by progressively exposing them. This would be akin to having found explicit values in system identification.

This implicit system identification via curriculum ensures the agent's internal representations progressively build fidelity, without ever fitting a parametric model. This process echoes, in spirit, ideas from system identification. The curriculum acts as a scaffold for building intuitive control before the agent faces more challenging dynamics.

3

Related Work

3.1. Reinforcement Learning for Nonholonomic Robotic Control

RL has been applied to a variety of non-holonomic mobile robots (which include skid-steer and wheeled vehicles that cannot move sideways) with promising results, especially when combined with other techniques. However early work demonstrated that even without other techniques model-free RL can handle the complex wheel-ground interaction and nonholonomic constraints of a skid-steer system. As long as the learning process is properly setup. Srikonda et al. (2022) used deep RL (specifically DDPG) to learn control of a skid-steer robot for trajectory tracking. In their setup, the agent controlled the robot's angular velocity to follow reference trajectories. With appropriate state representation and reward shaping, the learned policy was able to achieve accurate path following – within roughly half the vehicle's width of error – on a variety of test paths. The problem used angular velocity for the action space and an always applied constant linear velocity. Therefor the solution has intelligently abstracted away the dynamics and minimized the action space to one dimension. This greatly simplified the exploration problem and made a solution feasible without guidance.

In the specific case of skid-steer vehicles (differential drive or tracked vehicles), another key work is by Dai et al. (2022). While we cover it in more detail in Section 3.3 (as it involves expert knowledge integration), it's worth noting here that Dai et al. tackled exactly the problem of learning an optimal torque distribution for skid-steer velocity tracking. They used an RL agent (DDPG-based) to map state errors to wheel torques. Crucially, they identified that training this agent from scratch was extremely slow and unstable due to the reasons discussed earlier (complex dynamics, need for precise coordination between wheels). This motivates their knowledge-assisted approach (KA-DDPG) where they injected a form of expert policy into the learning loop. The success of Dai et al. (2022)'s agent in eventually achieving accurate velocity tracking on a skid-steer platform demonstrates that, given the right support, RL can handle non-holonomic torque control tasks. Their results showed near-optimal tracking after training, bridging towards classical control performance but achieved through learned policy.

Beyond skid-steer, non-holonomic vehicles such as car-like robots (Ackermann steering) and quadrupeds have been studied with RL. Many have found that some assistance (be it demonstrations, or curriculum) is needed to get reliable convergence. Han et al. (2025) and Datar et al. (2024) both tackle RL for Ackermann-steered vehicles in complex environments, and they report that a naïve end-to-end RL agent achieved almost no success on the task without additional help. In their experiments, a RL agent attempting to learn high-level driving in cluttered or rough terrains failed to converge to a reasonable policy when starting from random exploration. Both papers show that imitation-style learning from offline datasets is necessary to be successful.

Similarly, Wiberg et al. (2021) showed that a 16-ton forestry machine with six wheels, a steering mechanism, and active suspensions could learn to drive through challenging outdoor terrain. This proves even high degree of freedom nonholonomic vehicles can be trained via RL. In their setup, the agent's task was to reach target waypoints in a rough forest environment. Notably, they found it critical to use a combination of sparse and dense rewards along with a carefully structured training regime to handle

the vast state space. The agent was eventually able to handle mud, slopes, and obstacles, indicating that model-free RL can handle non-holonomic dynamics given enough training and a thoughtful reward design.

A recurring theme is that task design and additional training structure are crucial for success in these systems. Wiberg et al. incorporated domain knowledge by progressively exposing the agent to more difficult terrains (effectively an integrated curriculum, discussed in Section 3.2) and by penalizing unsafe behavior.

Other studies have addressed non-holonomic control by simplifying the learning problem or using hybrid approaches. An early example is the work of Ostafew et al. (2014), who blended learning with classical control for skid-steer path tracking. They used a nonlinear model-predictive controller (NMPC) as an expert policy and allowed the robot to learn corrections to this controller from experience. After each trial, the deviation between the expected and actual trajectory (caused by unmodeled slip and terrain effects) was used to update a learned offset in the controller. This can be seen as a form of iterative learning control or residual RL, where the base policy is a physics model and the learning component compensates for dynamics not captured by the model. Ostafew et al. (2014)'s approach highlights that incorporating domain knowledge (here, a physics model/controller) can significantly improve learning outcomes on nonholonomic robots – the learned policy improved the robot's ability to follow paths on outdoor terrain beyond what the NMPC alone could do. In summary, related work indicates that RL for non-holonomic vehicles is feasible and can produce robust controllers, but typically only when combined with strategies to mitigate the exploration difficulty. This sets the stage for approaches that intentionally incorporate domain knowledge into the RL training process, which we explore in the next sections.

3.2. Curriculum Learning in Robotics

Curriculum learning has been widely applied in robotics as a solution for tasks that are too difficult to learn in one go. The concept is to start with simpler versions of the task and progressively increase difficulty as the agent's proficiency improves. The works of Wiberg et al. (2021) and Xu et al. (2024) are illustrative case studies of curriculum learning, particularly relevant to our domain. As mentioned,Wiberg et al. (2021) trained a heavy skid-steer forestry vehicle using a staged approach. In their training, early episodes featured mostly flat, nominal terrain and strong incentives in the reward to avoid slip or instability. This allowed the agent to learn basic driving and not tip over. As training progressed, the environment was made progressively harder: larger logs on the ground, steeper inclines, and more uneven terrain were introduced. By the time the agent encountered these extreme conditions, it had already mastered basic locomotion and balance. The gradual exposure functioned as a curriculum that enabled the policy to eventually handle terrain that would have been impossible to learn on from scratch. Notably, if they attempted to train the agent on the hardest terrain from the very beginning, it failed consistently (the robot would tip or get stuck, yielding near-zero reward and no learning signal) . Their results clearly show that the curriculum was necessary: it improved not just training stability but also the final capability of the agent (which in the end could traverse obstacles that a non-curriculum agent never managed).

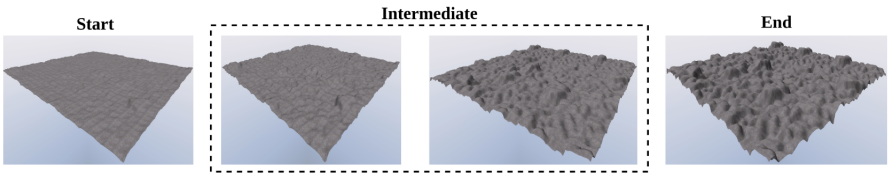


Figure 3.1: Increasing Mobility Difficulty of Vertically Challenging Terrain by Interpolating Start and End with a Weight (Xu et al., 2024)

Xu et al. (2024) provide another example. Xu et al. (2024) addressed autonomous off-road navigation on very steep and bumpy terrain. They used an RL agent to control an Ackermann-steered vehicle (action space: linear velocity and steering angle). Like Wiberg et al. (2021), they explicitly designed a curriculum for the terrain difficulty. Initially, the agent was trained on gentle slopes and smooth surfaces. Once it performed well there, the terrain difficulty was increased: slopes became steeper and obstacles

larger (Figure 3.1. This process continued until the agent was facing the most challenging terrain in simulation. The reward function was crafted to encourage steady progress and heavily penalize unsafe behavior (excessive roll/pitch indicating near-tip-over). By the end, the policy trained with this terrain curriculum was capable of driving a real four-wheeled robot up slopes and over obstacles that would have been far beyond its initial abilities. The curriculum was key to this success – it allowed the agent to incrementally extend its competence. Xu et al. (2024) conclude that the curriculum yielded improved learning stability and a higher final skill level than direct training.

Beyond these, numerous other works have leveraged curricula. Hoeller et al. (2023) for quadrupedal parkour (discussed in Section 1.1) is a prime example in legged robots. Another example is the work by Rudin et al. (2021), who trained a quadruped robot to traverse challenging terrains in a matter of minutes by combining massively parallel simulations with an automatic curriculum strategy. They progressively adjusted the difficulty of the terrain as the agent's performance improved, ensuring that the agent was always slightly challenged but not overwhelmed. This automatic curriculum, along with thousands of parallel training instances, enabled rapid learning that would otherwise be unattainable.

Curricula have also been tailored to wheeled quadrupeds. Chamorro et al. (2024) extended curriculum learning to a hybrid wheeled-legged robot tasked with climbing stairs. Instead of training directly on full-sized stairs (which caused the agent to fail initially), they introduced the robot to smaller steps and gradually increased the obstacle height. Dynamics were abstracted by using position for the leg joints and velocities for the wheels. The policy learned how to approach and surmount steps incrementally, ultimately succeeding on a standard staircase. This is a clear demonstration that curriculum learning can be crucial for tasks involving discrete jumps in difficulty: by smoothing out the difficulty curve (in this case, step height), the agent can make continual progress.

Margolis et al. (2022) provide an instructive study on curriculum design. They systematically evaluated multiple curriculum strategies for an autonomous locomotion task, comparing how different sequences of increasing difficulty affected learning outcomes. In one of their experiments, they gradually raised the limits on linear and angular velocities that an RL agent was trained on, similarly to our approach. The curricula that incrementally expanded the velocity range led to significantly improved stability in training and higher final performance (See Figure 3.2), compared to trying to train on the full velocity range from the start.Margolis et al. (2022)'s work is particularly important because it highlights that not all curricula are equally. They demonstrate that a curriculum's effectiveness hinges on modeling multi-dimensional task parameters jointly rather than independently. Their Grid Adaptive approach unlocks adjacent cells in the joint velocity–turning grid once a success threshold is met. It covers the feasible command space more thoroughly and yields lower tracking errors than a simpler "Box" curriculum. The "Box" curriculum grows the problem dimensions in isolation, meaning the linear and angular commanded velocities while coupled in there complexity are explored seperately. Because high linear speeds reduce feasible turning rates, modeling their interaction in one grid lets the curriculum avoid impossible tasks and instead expand along the true feasible boundary. By aligning the sampling distribution's shape and growth rules with the true difficulty frontier of the robot's dynamics, the Grid Adaptive curriculum substantially outperforms a simpler Box-style schedule as nicely visualised in the heatmap of Figure 3.2.

In summary, curriculum learning has proven to be a key enabler for training RL agents on complex robotic tasks. By decomposing a hard problem into a sequence of achievable steps, curricula allow the agent to build up competency gradually. The related work we surveyed, from legged locomotion to wheeled navigation, shows a consistent trend: agents trained with a curriculum outperform those trained on the hardest task from the outset, in terms of both learning speed and robustness of the final policy. This justifies our incorporation of curriculum learning in training a skid-steer robot: we expect that structuring the learning process (e.g., starting with low-speed trajectory tracking and ramping up to high speeds) will yield better results than a one-shot learning attempt. Notably, while curricula and demonstrations are often studied separately, there is growing interest in how they can be combined, which we discuss in Section 3.4.

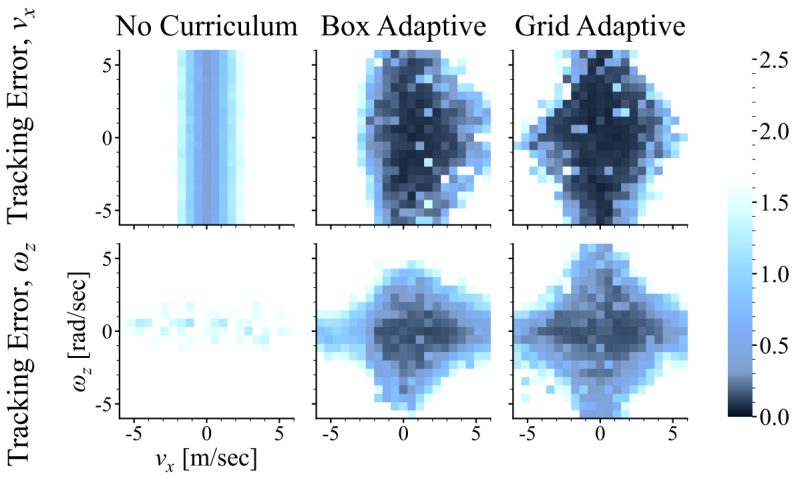


Figure 3.2: Heatmap of converged tracking error for curricular strategies (Margolis et al., 2022)

3.3. Imitation Learning in Robotic RL

Integrating expert demonstrations or controllers into the RL process – what we refer to as knowledge-assisted RL – has been explored in various forms. The core idea is that an expert (either a human operator or a well-tuned autonomous policy) can provide examples of good behavior, which the learning agent can use to bootstrap or guide its policy. This has been applied in various forms, from straightforward pre-training of a policy network on demonstration data to more integrated approaches where the expert influences the agent during RL training through action injection (Xie et al., 2022). Several key works illustrate different ways to combine imitation and RL.

The earlier example of Ostafew et al. (2014) is useful here to set the range of IL methods. This approach can be seen as a form of iterative learning control or policy improvement where the expert provides a strong initial policy and the learning refines it. This is a relatively conservative form of demonstration integration: the expert is always active (no pure exploration by the agent), ensuring safety but limiting the scope of changes to incremental improvements.

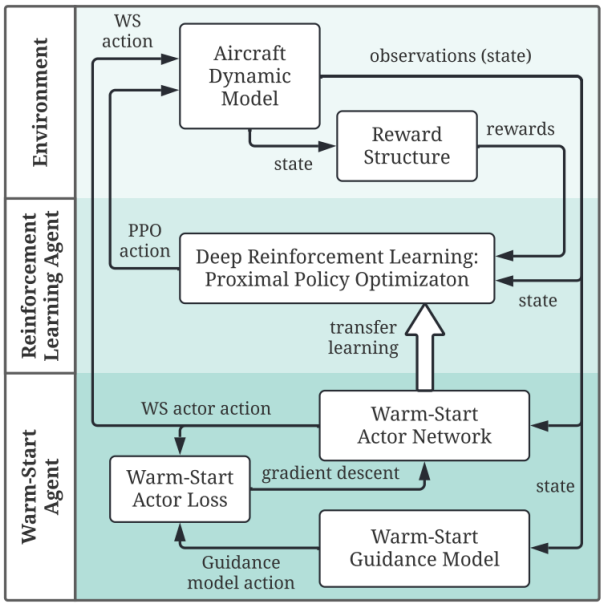


Figure 3.3: Agent-environment configuration for the flight guidance problem with RL and warm-starting (Coletti et al., 2023)

Another approach is to use IL for policy initialization. Instead of starting with a random policy, the agent is first trained (via supervised learning) to imitate the expert, and then this pre-trained policy is used as the starting point for RL fine-tuning. Coletti et al. (2023) provide a compelling example of the benefits of this strategy. They addressed a challenging flight control task for a fixed-wing unmanned aerial vehicle (a nonholonomic system with complex dynamics) by first cloning a classical guidance law through imitation. After this imitation phase, they switched to RL for fine-tuning the policy (See Figure 3.3). The difference in outcomes was dramatic: the RL agent initialized with the expert's behavior achieved over a 57% success rate in hitting waypoints, whereas an agent learning from scratch barely reached 2% success. This warm-start via IL essentially solved the poor exploration problem – the agent began near a good local policy and RL could then improve it further.

Likewise, in the context of ground robots, Han et al. (2025) demonstrated that pre-training a wheeled robot's policy on a dataset of human teleoperation and classical controller trajectories significantly improved its performance when subsequently fine-tuned with RL. In their experiments on a low-cost robot platform, policies that did not receive demonstration data failed to learn effective navigation, while those that were warm-started with IL not only learned faster but also transferred better from simulation to the real robot. These studies confirm that imitation learning can provide the agent with essential "driving lessons," so to speak, allowing it to handle the basics of control and thus giving RL a fighting chance to succeed on complicated tasks. Their work is aligned with findings from Hu et al. (2023) that emphasize combining model-based priors, large datasets, and RL as a recipe toward general-purpose robot learning.

Another class of approaches blends IL and RL more continuously. Rather than a single pre-training stage, the expert's guidance is offered throughout the RL training process. Dai et al. (2022) introduced a knowledge-assisted DDPG (KA-DDPG) algorithm for exactly our problem domain: learning an optimal torque distribution for a skid-steer vehicle. Instead of providing full demonstration trajectories, their method imbues the RL agent with expert knowledge in two ways: (1) blending the agent's action with an expert's action (especially early in training) to prevent catastrophic exploration, and (2) augmenting the reward with an extra term (a "guiding reward") when the agent's action aligns with what the expert would do (see Figure 3.4). In their implementation, a low-fidelity classical controller is used as the expert to suggest reasonable torques; with a certain ratio, this suggestion overrides part of the agent's action, and the agent also gets bonus rewards when it chooses a torque similar to the expert's suggestion. This approach can be seen as a form of on-line imitation: the agent is still primarily doing RL, but the expert nudges it away from bad regions of the action space and towards good actions. Dai et al. (2022) reported that knowledge-assisted RL substantially accelerated convergence in the skid-steer velocity-tracking task compared to standard RL, underlining the value of expert guidance even when it is not used to initialize the policy outright.

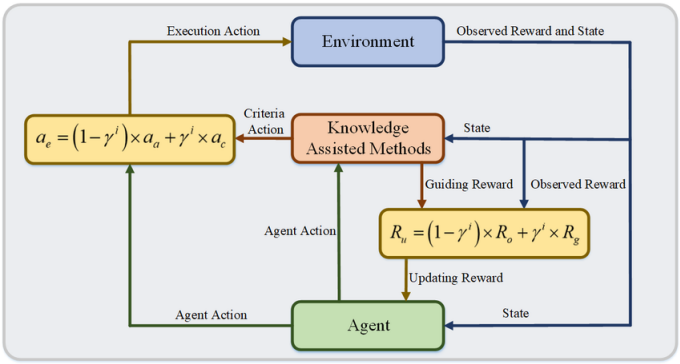


Figure 3.4: Framework of KA-DDPG (Dai et al., 2022)

In effect, early in training the agent is heavily incentivized to do exactly what the expert would do (both via action blending and reward), and later in training these influences taper off, letting the agent optimize the true task. The ratio changes over time to favour the policy actions more as time progresses. This method is quite aligned with our proposed KAMMA approach (Section 4.3), the difference being that KA-DDPG blends actions while we will propose a probabilistic switch. Dai et al. (2022) reported that KA-

DDPG learned a near-optimal torque controller much faster than standard RL. The agent avoided the random wandering phase and never exhibited the excessive wheel slip that a pure RL agent did at the start. By the end, when the expert's influence was gone, the agent had learned a policy that matched the expert in easy conditions and even exceeded it in some cases (because the RL fine-tuning can, in principle, discover better torque allocations than the simple expert controller). This demonstrates the power of expert-guided exploration, it is a central inspiration for our baseline and method.

Yet another technique is to impose expert knowledge as constraints. Tsampazis et al. (2023) exemplify this by using action masking based on heuristic rules during RL training. In their work, an autonomous off-road vehicle's policy was trained with certain actions forbidden if they were clearly unsafe or ineffective according to prior knowledge (for instance, the agent was not allowed to accelerate forward if an obstacle was immediately in front of the robot). By disallowing obviously bad actions, the agent's exploration space was reduced to more plausible behaviors, effectively injecting human expertise in a negative sense (preventing foolish actions rather than explicitly demonstrating good ones). This idea, while not imitation in the classical sense, shares the same spirit: leveraging expert understanding to shape the agent's behavior. To synthesize, related work on demonstration-enhanced RL shows two broad strategies:

- **Concurrent guidance:** expert intervention during training (like Ostafew's residual learning or Dai's KA-DDPG).
- **Sequential bootstrapping:** expert used in a pre-training phase or for initialization (like Han's and Coletti's approaches).

Our work leans on the concurrent guidance strategy: we will use an expert policy during RL training to influence the agent, rather than only pre-training and releasing. The novelty we seek is to refine how this guidance is applied (via KAMMA's switching strategy) and to combine it with the curriculum concept from Section **??**.

3.4. Combining Imitation Learning and Curriculum Learning

Both imitation learning and curriculum learning have individually been shown to greatly benefit policy learning in robotics. A natural question is whether combining these two approaches can provide complementary advantages. Intuitively, imitation learning addresses the initialization and early training phase, while curriculum learning sustains progress by gradually increasing difficulty. If used together, an agent could start with a good initial policy and always train on an appropriate level of challenge. Some recent works have begun to explore this interplay, although it remains relatively underexplored compared to the standalone use of IL or curricula. Chivkula et al. (2022) used the process show in Figure 3.5 below to kickstart their curriculum.

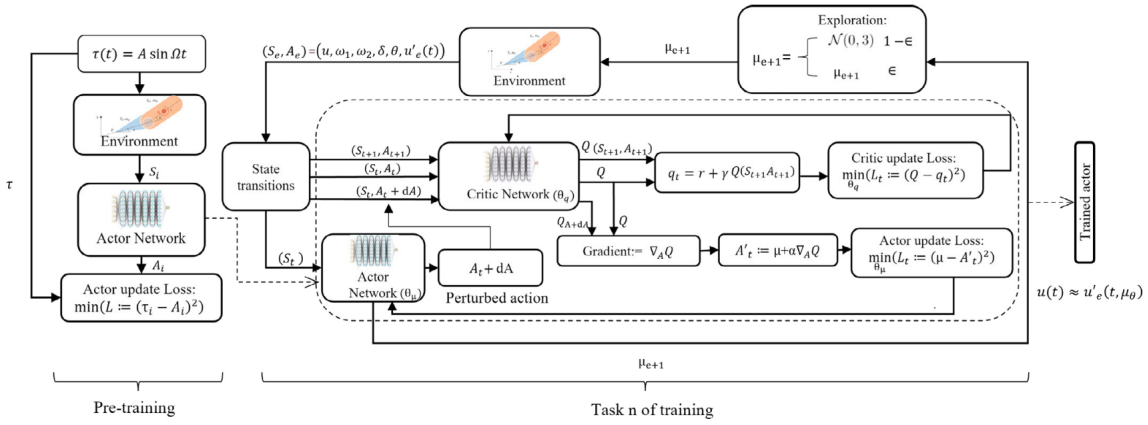


Figure 3.5: Here we illustrate the training process. On the left, the DDPG actor is trained to mimic a given $\tau(t)$. In the center, the actor is updated over multiple episodes of DDPG, where the target velocity function u′e(t) is changed between episodes according to the curriculum, resulting in a fully-trained actor capable of simultaneous path and velocity tracking (Chivkula et al., 2022)

Chivkula et al. (2022) also takes advantage of sequential bootstrapping and offers a clear example of a two-stage training process that embodies both concepts. In their work, they focused on an under-actuated nonholonomic system (a Chaplygin sleigh) and employed a form of self-imitation across a curriculum of two tasks. In the first stage, they trained a policy through RL to perform a simple cyclic movement (essentially driving the system in repetitive limit cycles). This first-stage policy was far from the final objective but captured a basic gait for the robot (see Figure 3.6 (1a), and (1b)). In the second stage, they tasked the agent with more complex trajectory tracking descriptions (see Figure 3.6 (2a), (2b), and (2c)) and used the first policy as an expert demonstrator to guide the learning of the second policy. In practice, the trajectories generated by the first policy (the simple cyclic motion) were treated as demonstration data for the second stage, a technique sometimes called learning by bootstrapping. Their results showed that the knowledge transferred from the initial policy significantly accelerated the training of the final policy – even though the initial gait was suboptimal for the new task, it provided a helpful scaffold for learning the more complex maneuver. This approach can be seen as a curriculum (stage 1 was easier, stage 2 harder) combined with imitation (stage 2 imitates the policy from stage 1). It validates the idea that a policy learned on a simpler task can serve as a form of expert for a related harder task, thus combining the benefits of prior learning and task progression.

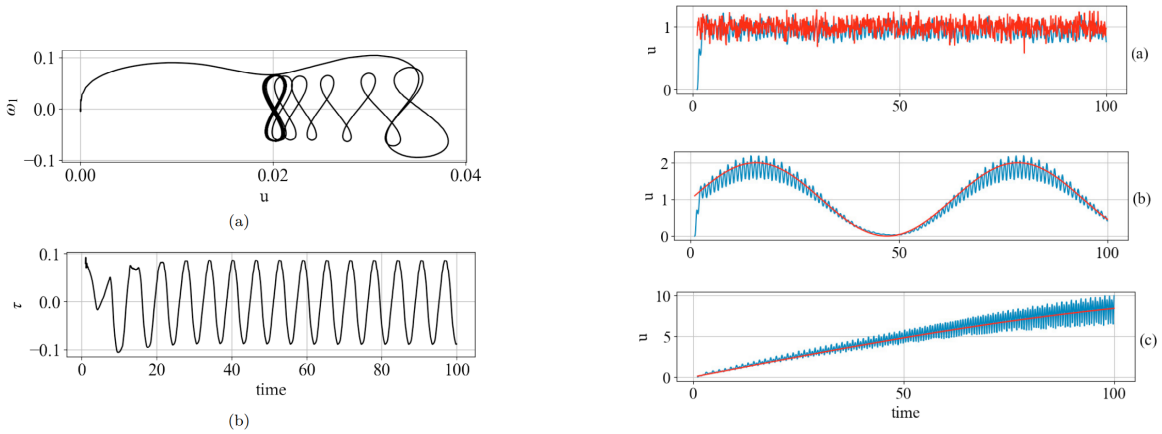


Figure 3.6: These two images show the results of the pre-traing phase on the left, and the curriculum design and on the right. We see a slow convergence to the prescribed limit cycle in (1a) and the prescribed actions in (1b). The training curriculum of phase 2 shows the target velocity for each time interval. In (2a) the velocity is sampled from a normal distribution u′(t) $(1, \sigma)$, where σ increases be-tween iterations. In (2b) the agent learns a policy to track a sinusoidal u′(t) with increasing amplitude and frequency. In (2c) the agent learns to track monotonically increasing velocities to a maximum of u′(t) = 10. (Chivkula et al., 2022)

In a sense, the method of Coletti et al. (2023) can also be explored differently: they first used IL on an easier behavioral cloning task (imitate an existing controller), then used RL on the harder task (improve beyond the expert), which is a sequential combination of IL and RL that resembles a curriculum of training modes. However, in Coletti et al. (2023)'s case the "curriculum" was not about environment difficulty but about learning method – nevertheless, it underscores that IL and RL can be arranged in stages to yield better performance than either alone.

Despite these examples, there is still a lack of systematic studies on combining curricula with external expert demonstrations in one unified framework. Most literature either focuses on curriculum learning assuming the agent starts from scratch, or focuses on imitation learning and perhaps uses a trivial task switch (from imitation phase to RL phase). Very few works explicitly design a multi-phase curriculum and integrate expert demonstrations throughout those phases. The potential synergy is clear: an expert-provided policy could be used at each curriculum step to guide the agent, or conversely, a curriculum could be designed to maximally exploit a given set of demonstrations. In our view, this joint approach is a promising yet underexplored direction.

3.5. Synthesis and Research Gap

From the above review, we can draw a few important conclusions. First, curriculum learning (Section 3.2) and expert-assisted RL (Section 3.3) have independently proven successful in tackling complex

control problems. Curricula handle the task complexity by staging the challenges, and demonstrations handle the exploration difficulty by providing initial policy knowledge or ongoing guidance. Both lead to more efficient and stable learning processes.

However, there is a clear gap in combining these two dimensions. To our knowledge, no prior work has systematically evaluated the impact of using curriculum learning and expert demonstrations together in a unified framework for RL. In other words, we lack studies where an agent is simultaneously aided by an expert and trained across multiple stages of increasing difficulty.Margolis et al. (2022) and others have compared different curriculum strategies (without expert assistance), and works likeDai et al. (2022) have demonstrated knowledge-assisted RL on a fixed task distribution (without curricula). But we have not seen an analysis of whether expert guidance remains effective across curriculum stages or how curricula might need to be designed differently when an expert is in the loop.

This is the niche our research aims to fill. The core research gap can be summarized as: How to best integrate human/heuristic expertise with incremental curriculum learning for training deep RL agents in complex control tasks. We hypothesize that these techniques are complementary and that together they will yield superior results to either alone. For instance, an expert might help the agent clear early curriculum stages faster or with less variance, and the curriculum might allow the expert's influence to be gradually reduced in a controlled manner as the agent proves competence at each stage.

In this thesis, we address the above gap by integrating curriculum learning with imitation/ expert guidance for torque-based skid-steer control. We design and compare distinct curricula (different ways of gradually increasing task difficulty) and evaluate each by comparing to an IL baseline. By comparing across these scenarios, we can assess whether the combination (IL + curriculum) indeed provides synergy in terms of data efficiency and final performance. Our expectation, based on the motivations outlined, is that an agent trained with both techniques will learn faster and achieve more robust control than an agent with either technique alone.

Moreover, our approach directly builds on insights from the related work. We extend the idea of Margolis et al. (comparing curricula) by adding the dimension of expert assistance, and we extend Dai et al.'s knowledge-assisted learning by applying it across a curriculum of tasks. In doing so, we aim to provide a deeper understanding of how to best combine human expertise with curriculum learning in RL.

In summary, the literature suggests strong individual merits for using curricula and for using demonstrations in RL. The intersection of these two – a curriculum-guided, expert-augmented RL training regime – remains under-explored. This thesis seeks to contribute to that intersection, showing how a knowledge-assisted RL agent can progress through staged tasks to master a problem that neither pure curriculum nor pure expert-guided RL could solve as efficiently on their own.

4

Methodology: From KA-DDPG to KAMMA + Curriculum

4.1. Problem Formulation (Action, Observation, Reward)

We consider a skid-steer mobile robot learning to follow a target velocity. For our research we have focused the task of linear velocity only, so during all experiments the angular velocity command will be set to zero. This simplifies the task by taking away one degree of freedom to explore and makes the experiments more focused on the methods than the results. The typical skid-steer challenges also remain the same. Formally, at each timestep the agent (the robot's controller) observes the state, takes an action (wheel torques), and receives a reward. We define these elements as follows:

1. **Action Space:** The agent's action is the torque command to each of the robot's four drive motors (two on each side for a four-wheeled skid-steer). Thus, the natural action space is $\mathbb{R}^4$, where each component is the torque for one wheel. This 4D individual wheel torque control gives the agent full fine-grained control over all drive dynamics. As discussed, we will also evaluate reduced-dimensional action spaces for comparison. In a 2D torque action space, the agent outputs two values: one for the left-side wheels and one for the right-side wheels (each value is applied to both wheels on that side). This grouping simplifies the action space by treating each side as a unit – it removes the differential control between wheels on the same side, which might be acceptable if those wheels experience similar conditions. In our simulation we test on flat ground and thus have those similar conditions. We therefor expect the reduced dimensionality to perform better than the 4D action space. To fully test this we also experiment with a 1D torque action space. For this reduced action space we also expect better performance than the 2D variant. The primary action space for our method remains 4D torques, as it provides the highest level of control expressivity required for complex non-linear dynamic terrain interactions.

2. **Observation Space:** The agent's observation is designed to capture the tracking error and system state relevant to control. We include the longitudinal (forward) velocity error $e_v = v_{\text{desired}} - v_{\text{actual}}$ and the yaw (angular) error $e_\omega = \omega_{\text{desired}} - \omega_{\text{actual}}$. These represent how far off the robot is from the commanded trajectory at a given moment. In addition, we include the derivatives of these errors, e'_v and e'_ω, which correspond to the robot's longitudinal acceleration and yaw acceleration. Including the rate of change of error helps the agent infer whether it is converging to or diverging from the target velocity, which is useful for damping and stability. We deliberately omit lateral velocity or slip measurements from the state, except insofar as they affect yaw dynamics; skid-steer robots often exhibit lateral slip, but that is not directly controllable and can introduce noise. By focusing the observation on longitudinal and yaw dynamics (which are the controllable degrees of freedom), we align the state with the key control objectives and avoid distracting the agent with variables it cannot directly regulate. In addition to errors, the observations also include the current commanded velocity. This is important because without this commanded velocity state aliasing can occur where different traction regimes can not be separated or inferred. If the ob-

servations only see a relative error, guiding the system through a curriculum based on absolute speeds cannot learn anything meaningful as it will be generalizing a policy over the entire velocity range. We use the commanded velocity and not the actual velocity as this is a noise free signal and through the velocity errors the actual velocity is also part of the observations. In actuality noise in the observations is inevitable, either the current velocity or the velocity error will create noise in the signal. We further do not provide direct information about the terrain or friction; the agent must infer those through trial and error or implicit system identification.

3. **Reward Function:** The reward is structured to strongly encourage accurate trajectory following while discouraging excessive control effort or unstable motions. We design a dense reward with multiple components:

   - **Velocity Tracking Reward:** A primary term r_{track} that is high when the robot's actual linear and angular velocities match the desired values, and low when there is a large error. For example, one can use $r_{\text{track}} = -(|e_v| + k \cdot |e_\omega|)$ for some weight k, so the reward is the negative of the sum of absolute errors (or squared errors) in forward and angular velocity. This term continuously guides the agent to minimize tracking error.
   - **Smoothness Penalty:** To avoid erratic control, we add a penalty on abrupt changes in action or on high accelerations. In practice, using the error derivatives, we include a term like $r_{\text{smooth}} = -\left(e_v'^2 + e_\omega'^2\right)$ which penalizes rapid changes in velocity error equivalently large accelerations. This term indirectly penalizes jerky torque commands and slip, because large sudden torques would cause spikes in acceleration errors.
   - **Control Effort Penalty:** We also penalize excessive wheel slip or torque usage. This can be implemented as a penalty on the magnitude of applied wheel torques. This is implemented as $r_{\text{torque}} = -(T_{fl}^2 + T_{rl}^2 + T_{fr}^2 + T_{rr}^2)$ In our reward, this is partly covered by the smoothness term on error derivatives and partly by the tracking reward. If slip is detected e.g., if the linear velocity is not increasing despite torque, indicating slipping, the error remains high which already yields low reward.
   - **Success Bonus:** Finally we include a sparse bonus r_{success} for excellent performance: if the robot gets the velocity error within a tight bound, it gets an additional reward. Every time step that $e_v| < \epsilon_v$ and $|e_\omega| < \epsilon_\omega$ (errors within acceptable limits), we give a small positive bonus (e.g., +0.2). This incentivizes not just instantaneous error reduction but maintaining accurate tracking over time. By setting this limit below the tracking performance of the controller you also extra incentivize improving on the controller.

   The total reward at time t can be formulated as: $r(t) = r_{\text{track}}(t) + r_{\text{smooth}}(t) + r_{\text{torque}}(t) + r_{\text{succes}}(t)$, where weightings can be applied to each component for reward shaping. This reward provides continuous feedback guiding the agent to minimize tracking error smoothly. During training, the agent thus learns to both be accurate and gentle: it gets highest reward for matching the commanded speeds with minimal oscillations or wheel spinning.

With the above formulation, the learning task is an MDP where the agent observes the velocity tracking errors, applies motor torques, and is rewarded for minimizing those errors without excessive control. The state-space is relatively low-dimensional, and the action-space is effort based, continuous, and potentially high-dimensional (4D). This formulation is challenging due to the complex and nonlinear dynamics linking actions to future states (torque to next velocity and position). But it is also well-defined for applying actor–critic RL methods like DDPG.

4.2. KA-DDPG: Baseline Implementation and Limitations

As a baseline knowledge-assisted approach, we implement the Knowledge-Assisted Deep Deterministic Policy Gradient (KA-DDPG) algorithm inspired by Dai et al. (2022). The KA-DDPG framework augments the standard DDPG agent with two forms of expert-derived assistance during training: action guidance and reward shaping. Specifically, we incorporate a low-fidelity expert controller (providing what will be called criteria actions) and an evaluation module (providing the guiding rewards) into the training loop. The expert controller is a simplified or suboptimal policy that can generate reasonable actions for the task, albeit with limited accuracy. In our application, the P-controller is defined as in

equation 4.1 below. Here J is the vehicle inertia around it's own center of gravity, m is the mass of the vehicle (see Table 4.1) and K_p is the proportional constant.

$$\begin{aligned} T_{lf} = T_{lr} &= K_p \left(m \cdot \frac{v_\Delta}{dt} - J \cdot \frac{\omega_\Delta}{dt} \right), \\ T_{rf} = T_{rr} &= K_p \left(m \cdot \frac{v_\Delta}{dt} + J \cdot \frac{\omega_\Delta}{dt} \right), \end{aligned} \tag{4.1}$$

A second motivation to go for sub-optimal pure–P rather than PI control is: the integral term depends on accumulated past errors, which conflicts with the Markov assumption in our RL setup (where updates only see the current state), and it would further blur credit assignment by hiding which moments of error drove the correction. By using a memoryless P controller, we ensure the expert's outputs remain compatible with the agent's MDP-based learning and keep the learning signal clear.

While the injected hand-crafted policy cannot perfectly track arbitrary trajectories, it performs significantly better than random actions and thus can prevent the agent from floundering in the early learning phase. The evaluation module encodes additional domain knowledge about what constitutes good performance, beyond the basic environment reward. It produces an auxiliary reward signal that "guides" the agent toward expert-desired behaviors. Together, these two knowledge-assisted elements modify how actions are chosen and how rewards are computed during training, as described next.

1. **Criteria Action Blending (Expert Action Injection):** Figure 4.1 shows the KA-DDPG framework, at each decision step the agent not only computes its action from the actor network, but also receives an alternative action suggestion from the expert controller (Dai et al. (2022)). Rather than executing the agent's action directly, the algorithm combines the agent's action with the criteria action to determine the actual control input sent to the environment. The core idea is to let the expert policy guide the agent, especially when the agent is still near-inexperienced. There are multiple ways to realize this combination; a conceptually simple approach is to use a weighted blend. The executed action is defined at time t as:

$$a_{exec} = \alpha(t) * a_{critic} + (1 - \alpha(t)) a_{agent}, \tag{4.2}$$

   Where a_t^{crit} is the expert's criteria action and a_t^{agent} is the RL agent's own action output at state s_t. The factor $\alpha(t) \in [0, 1]$ controls the influence of the expert. At the beginning of training, we set $\alpha(t)$ high (near 1), meaning the agent largely follows the expert's decisions (acting as if it were imitating the expert). This ensures that the vehicle's behavior stays in a reasonable regime and achieves some minimal level of performance, thereby reducing the incidence of very poor (random) actions early on. As training progresses, $\alpha(t)$ is gradually decayed toward 0 over episodes. For example, α can be geometrically decayed, or scheduled to decrease in stages. Eventually, $\alpha(t) \approx 0$, and thus $a_{exec}(t) \approx a_{agent}(t)$ – the agent acts according to its own policy with no interference. In effect, this criteria action replacement strategy starts training in a teacher-student mode, where the expert (teacher) heavily influences actions, and then transitions to full autonomy as the student gains proficiency. By episode end, the agent has learned to select reasonable actions learning from the augmented reward, but it got there with far more guidance

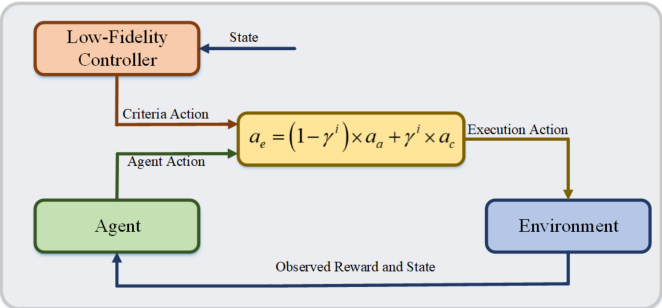


Figure 4.1: Criteria action method (Dai et al., 2022)

than a conventional agent. This approach speeds up exploration by focusing on actions that are known to be better than random, and it also contributes to safer learning since the agent is less likely to take extreme actions that could cause instability or failure.

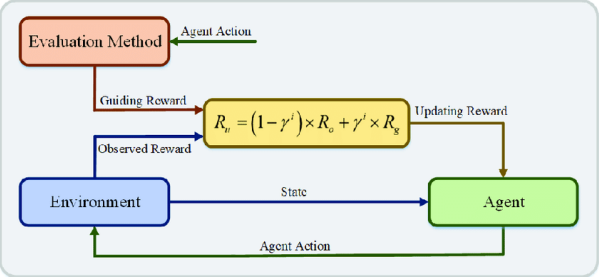


Figure 4.2: Criteria reward method (Dai et al., 2022)

2. **Guiding Reward Augmentation (Reward Shaping with Expert):** Figure 4.2 shows the second knowledge-assisted mechanism addressing the reward signal. In a plain DDPG setup, the agent updates its policy using the observed reward provided by the environment at each step (for example, a reward might be a negative tracking error minus some torque penalty). KA-DDPG instead computes an augmented reward for training by adding an extra term, the guiding reward, derived from the expert's evaluation. The purpose of this guiding reward is to "sharpen" or bias the feedback such that actions more aligned with expert knowledge receive higher reinforcement. A simple yet effective choice for the guiding reward is to use the discrepancy between the agent's action and the expert's action as a basis. For instance, one can define the guiding reward r_{guide} at time t as the negative distance between the agent's torque command and the expert's torque command:

$$r_{guide}(t) = \|a_{agent}(t) - a_{critic}(t)\| \tag{4.3}$$

(where the norm can be Euclidean or another suitable measure). This r_{guide} is higher (less negative) when the agent's chosen action is closer to the expert's action, and it worsens (more negative) as the agent deviates from the expert. We then combine the environment's reward $r_{obs}(t)$ and the guiding reward to form the update reward used for learning:

$$r_{update}(t) = \beta(t) * r_{guide}(t) + (1 - \beta(t))r_{obs}(t), \tag{4.4}$$

Here $\beta(t)$ (analogous to $\alpha(t)$, and γ from Dai et al. (2022)) controls the influence of the guiding reward. Starting with $\beta(t) = 1$, means that early in training the agent is almost entirely rewarded based on how closely it mimics the expert. As training progresses $\beta(t)$is gradually decreased to 0, transitioning the agent's objective back to the true environment reward. Eventually, the agent is evaluated purely on the actual tracking task performance, once it has learned the basics from the expert.

Using both mechanisms, the KA-DDPG framework "warm-starts" the learning process. Early on, the agent's actions are largely guided by the expert (through action injection) and its learning signal is dominated by expert-based reward shaping. This approach provides heavy guidance initially: the agent essentially operates in a coach-led mode and avoids the worst actions while it learns. As competence increases, the guidance is tapered off, allowing the agent to fine-tune and even surpass the expert. By training in this manner, the agent can achieve reasonable performance from the very first episodes (due to expert actions) and improve steadily as it gains autonomy. In summary, the knowledge-assisted framework integrates a low-fidelity expert's actions and evaluations into the RL loop, aiming to improve learning speed and stability compared to a naive RL agent.

4.2.1. KA-DDPG: Limitations

We implement the above KA-DDPG approach in our skid-steer RL task. Empirically, this baseline drastically improves learning speed and stability compared to a naive RL agent. However, we identified

several limitations of the continuous blending strategy that motivate a refinement (like the instability shown in Section 5.2.1 (where the naive 4D agent failed to converge):

- **Blended Actions and Distorted Learning Signals:** The continuous interpolation between expert and agent actions in KA-DDPG introduces a fundamental ambiguity into the learning process. During early training, executed actions may be composed of, for example, 70% expert and 30% agent contributions, with reward signals similarly blended between the guiding and observed components. As a result, the agent is seldom exposed to the full consequences of its own policy decisions, since the expert dominates both the behavior and the resulting rewards. This skews the critic's training distribution toward the expert's control envelope and limits the agent's exposure to suboptimal regions of the action space—regions that would be informative for robust value estimation. Furthermore, even when the agent selects poor actions, the expert's corrective influence may suppress negative outcomes, preventing the agent from learning appropriate credit assignments. This persistent coupling compromises the agent's ability to discern whether the reward signal results from its own decisions or the expert's influence. Consequently, the RL policy may converge toward a distorted gradient signal, biased by the ever-present expert. While the "soft landing" approach is designed to prevent catastrophic failures during early learning, it can paradoxically impede the emergence of a well-calibrated policy. To still create a diverse data-set the agent should focus learning on the edge of the IL schedule decreasing the expert influence as fast as possible. This keeps the learning frontier as challenging as possible and allows using the learning chaos to create diversity in the dataset. Thiswill enrich the critic generalization and improve long-term policy robustness. Ideally, this fast learning would receive feedback grounded in unfiltered consequences of its own actions for example through Action Mixing explained in Section 4.3.

- **Replay Buffer Staleness:** KA-DDPG stores every (s,a,r,s′) tuple—including those collected early in training under untrained, suboptimal policies—in a large replay buffer. Remember that in early training the reward signal is mainly based on the guiding reward which gives penalties for differences between the policy actions and expert actions. Over time, as the agent's policy improves, these outdated transitions (with their old misleading rewards) remain available and continue to be sampled. They pollute the learning signal, because the critic is repeatedly trained on experience generated by now-superseded policies. This "recycling" of stale, low-quality data slows convergence and can even push the agent back toward poor behaviors.

- **Torque Saturation in Early Training:** Despite the expert dampening the agent's actions, we observed that the standard blending approach often led to torque saturation in the early episodes. Specifically in our high-dimensional 4D torque task, we often observe periods of policy saturation: the agent's network outputs extreme torques for many consecutive episodes. Because the blending factor α_t decays over time, the policy gradually contributes more to the final executed action. When both expert and policy produce strong, conflicting torques, their blend can become erratic, and the expert's corrections themselves become "bad examples" as they start to compensate for eachother. The agent then collects tuples whose "guiding reward" encourages following a corrupted expert action. In effect, the very mechanism that was supposed to protect and teach the agent begins to mislead it.

In summary, KA-DDPG's blending scheme, while effective, can be improved. The agent's learning process could benefit from a clearer separation between expert-driven experience and agent-driven experience. This realization led us to propose an alternative knowledge assistance mechanism, KAMMA (Knowledge-Assisted Mixed Mode Actioning), which we introduce next. KAMMA addresses the above issues by replacing continuous action interpolation with a probabilistic switching strategy, giving the agent more distinct learning signals and eliminating the unintended side effects of blending.

4.3. KAMMA: Probabilistic Action-Selection Extension

To mitigate the drawbacks of continuous action blending identified above, we propose Knowledge-Assisted Mixed Mode Actioning (KAMMA) as an improved action guidance mechanism. An overview of how we approached this is shown in Algorithm 1. The core idea of KAMMA is to make the agent and expert contributions mode-based rather than blended. In practice, instead of combining the actions

at every time step, we choose either the expert's action or the agent's action to execute at each time step, based on a certain probability. This yields a clearer separation of control: at any given time, either the expert is fully in charge or the agent is fully in charge. KAMMA can be viewed as introducing an epsilon-greedy strategy in the action selection of the learning process. We define a time-varying probability π_{expert} (analogous to α_t in KA-DDPG) such that:

- With probability $\pi_{\text{expert}}(t)$, the expert's action is applied: $a(t) = a_{\text{expert}}(t)$.
- With probability $1 - \pi_{\text{expert}}$, the agent's action is applied: $a(t) = a_{\text{agent}}(t)$.

Early in training, $\pi_{\text{expert}}(0)$ is set close to 1 (e.g., 1.0 or 0.99), meaning almost all actions are from the expert. Over the course of training, π_{expert} is decayed toward 0, following a schedule similar in spirit to that of α_t before. For instance, we might exponentially decay π_{expert} such that after a certain number of episodes it drops to near 0. By the end of training, $\pi_{\text{expert}} \approx 0$ and thus the agent is acting on its own in almost all time steps. Importantly, the decay of π_{expert} can be coordinated with the decay of the reward shaping factor β_t. In our implementation, we used the same decay schedule for β_t as for π_{expert}, ensuring that both forms of guidance taper off together. This means early on we have both action and reward dominated by the expert, and gradually both influences diminish. One of the problems with KA-DDPG originated at the Replay Buffer (RB). This memory was large and kept tuples with expired reward shapes in the buffer for updates. On solution to deal with this problem is the implementation of a First In First Out (FIFO) RB. By limiting the available memory and deleting the oldest samples we can mitigate the effect of recycling old reward shapes. This proved useful for the 1D, and 2D KA-DDPG variants where it realised faster convergence. It did however not change the results for the 4D variant. As the size of the FIFO RB brings variance and the performance boost is less pronounced in KAMMA, we will not incorporate any FIFO RB until the curriculum experiments. And we only use it there in order to illustrate the effects of catastrophic forgetting when using curriculum learning (elaborated on in Section 4.4). Another solution is now that we can also change the guiding reward formulation to always be defined with the applied action in stead of the policy action. That way the reward in a sample will actually hold information on the action included in the tuple in stead of only the policy part of the blended action in the tuple. This makes the earlier samples in KAMMA more truthful and further dispels the need for the FIFO RB. Both the mixing mechanism and the reformulated guiding reward should result in a more homogeneous reward evolution for KAMMA and further minimize the risk of instability.

4.3.1. Mechanics of KAMMA:

Although KAMMA's hard-switching cleanly separates expert and agent actions, it brings its own quirks in the learning signal. In the very early episodes, almost every step uses the expert's torque, so the applied action and the expert action coincide exactly. Because our guiding reward is defined as the negative norm between the applied and expert torques, it vanishes whenever the expert acts. What remains is only a tiny fraction of the environment reward (since the mixing factor still heavily weights imitation). As a result, those first few thousand tuples carry almost zero training signal. They are safe examples of expert behavior, but with little to teach the critic. This is still more informative than the blended signal from KA-DDPG however.

To prevent these low-information transitions from endlessly cluttering the replay buffer, we could implement a cap on the size and employ the FIFO queue talked about earlier. As more informative, agent-driven transitions arrive, the oldest expert-only tuples are dropped. This ensures that once the agent begins to take control, the critic is trained largely on transitions that reflect real performance and useful imitation feedback, rather than on a backlog of near-zero-reward demonstrations. The near-zero-reward can also be turned into a more informative sample by also adding some gaussian exploration noise to the expert action, instead of only to the policy actions. These changes can be a fruitful addition to KAMMA, however in practice we have noticed that keeping the earlier transitions in the RB improves stability and convergence speed. To make a definitive choice a study should be done on the relationship of performance and the FIFO RB size. This is however not inside the scope of this thesis.

Once KAMMA's imitation mechanism weight has decayed enough for the agent's actions to be executed regularly, each transition will carry both an environment reward (measuring true performance) and a smaller guiding reward (nudging the agent toward expert behavior). This clear progression of expert-only, agent-guided, then purely agent autonomous, makes credit assignment transparent and avoids

Algorithm 1 KAMMA: Knowledge-Assisted Mixed Mode Actioning

Require: Critic network Q_θ, and actor network π_ϕ
Require: Target networks $\bar{\pi}_\phi$, $\bar{Q}_\theta$, replay buffer $\mathcal{D}$
Require: Expert-mixing schedule $\{\gamma_t\}_{t=0}^{T}$, exploration noise process $\mathcal{N}_t$

for episode $= 1$ to M **do**
  $s_0 \leftarrow$ env.reset()
  for $t = 0$ to T **do**
    $u \sim \mathcal{U}(0,1)$
    if $u < \gamma_t$ **then**
      $a_t \leftarrow \pi_E(s_t)$
    else
      $a_t \leftarrow \pi_\phi(s_t) + \mathcal{N}_t$
    end if
    Execute a_t, observe $(r_{\text{env},t}, s_{t+1})$
    $r_{\text{guide},t} \leftarrow -\|a_t - \pi_E(s_t)\|^2$
    $r_t \leftarrow (1-\gamma_t)\, r_{\text{env},t} + \gamma_t\, r_{\text{guide},t}$
    Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    Sample mini-batch $\{(s_i, a_i, r_i, s_i')\}_{i=1}^{B}$ from $\mathcal{D}$
    $y_i \leftarrow r_i + \gamma\, \bar{Q}_\theta\big(s_i', \bar{\pi}_\phi(s_i')\big)$
    Update critic by minimizing

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\big(Q_\theta(s_i, a_i) - y_i\big)^2$$

    Update actor using deterministic policy gradient:

$$\nabla_\phi J = \frac{1}{N}\sum_{i=1}^{N} \nabla_a Q_\theta(s_i, a)\big|_{a=\pi_\phi(s_i)}\, \nabla_\phi \pi_\phi(s_i)$$

    Soft-update targets:

$$\bar{\theta} \leftarrow \tau\,\theta + (1-\tau)\,\bar{\theta}, \quad \bar{\phi} \leftarrow \tau\,\phi + (1-\tau)\,\bar{\phi}$$

    $\gamma_t \leftarrow \gamma_{t+1}$
  end for
end for

the drawn-out, confusing middle phase that arises in KA-DDPG when expert and agent torques are continuously blended.

Despite the nuances we expect KAMMA to address the issues identified with KA-DDPG. It should preserve the benefits of expert guidance (fast initial learning, safety) while reducing the blending interference in the learning process. We hypothesize that KAMMA will achieve better final performance to the standard KA-DDPG, and with improved training stability and possibly faster convergence. KAMMA essentially provides a cleaner “knowledge assistance” modality and thus we adopt it as the default guidance method in the rest of our methodology.

Before moving to the experimental validation, we integrate one more concept into the methodology: curriculum learning. The next section describes how we incorporate two distinctly different gradual curricula into the KAMMA training process to tackle the full complexity of the task.

4.4. KAMMA + Curriculum: Gradual Difficulty Scheduling

While KAMMA focuses on how the expert guidance is applied at each time step, curriculum learning focuses on what tasks the agent is learning over the course of training. We define training curricula

that progressively increase the task difficulty. The difficulty is primarily characterized by the range of velocities (linear and angular) that the agent must track, as well as the deviation from the expert controller's comfort zone. We implement and compare two distinct curricula:

1. **Growing Variance (GV) Curriculum:** In this curriculum, the distribution of target velocities begins narrowly concentrated around a moderate speed (approximately 1.5 m/s, which is near the expert controller's nominal operating point) and then gradually broadens over the course of training (see Figure 4.3). Specifically, early in training the agent is exposed to trajectories with speeds mostly near 1.5 m/s (low variance around this value). As the agent masters these, the variance of the speed distribution is increased in stages, allowing a wider range of speeds both below and above 1.5 m/s. Over time, the agent encounters both very low speeds (near 1 m/s) and higher speeds (toward the controllers limits $\tilde{2}$.0 m/s). This staged increase ensures that the agent first learns to handle the moderate regime and then steadily generalizes to more extreme regimes. The growing variance curriculum is designed to incrementally challenge the agent while still taking advantage of the guiding examples early in training
2. **Bimodal Drift (BD) Curriculum:** In this curriculum, the training velocity distribution is intentionally bimodal, with two distinct peaks – one drifting below 1.5 m/s and one above 1.5 m/s (see Figure 4.3)– and relatively fewer training scenarios at speeds around 1.5 m/s. For example, the agent might frequently see tasks at 1.5 m/s, but once this is mastered we focus on improving the envelope to 1.0 m/s and 2.0 m/s. This means the agent alternates between practicing on easy, slow trajectories and very fast, challenging trajectories, with less exposure to the intermediate speeds. The idea is to force the agent to learn control at low and high extremes, thereby covering a broad range of behaviors, and then later verify if it can still interpolate its knowledge to the mid-range. This setup should be more sensitive to catastrophic forgetting, which should influence the final performance. However this performance can also be influenced by another aspect of this curriculum. The sampling provides a non-monotonic progression later in training: the task difficulty jumps between easier and harder regimes rather than smoothly increasing in both directions. This also results in a non-homogeneous dataset and can be the cause for instability.

As discussed for our research we have focused the task on linear velocity only, so during the experiments the angular velocity command will be set to zero. This setup should help observe the effect of catastrophic forgetting by better isolating for that problem. By only commanding a linear velocity, no extra difficulty will arise from combinations of the coupled linear and angular velocity space as was experienced by Margolis et al. (2022).

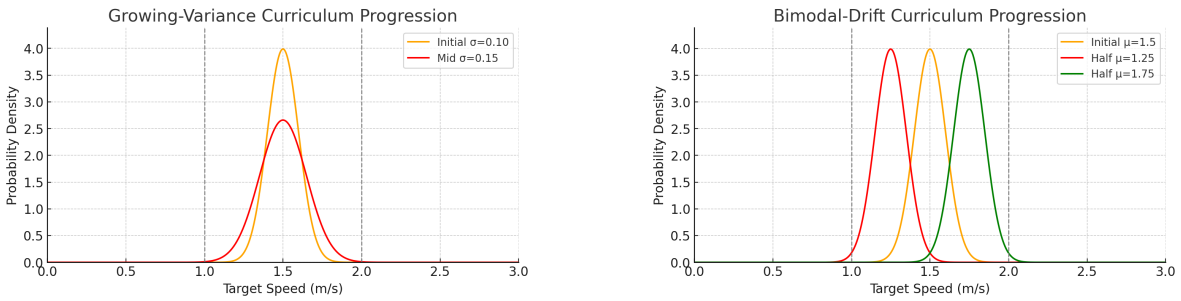


Figure 4.3: A schematic of the distributions belonging to the Growing Variance Curriculum (a), and the Bimodal Drift Curriculum (b)

4.5. Implementation Details and Hyperparameters

All training and experiments are conducted in a high-fidelity simulation to allow safe, fast, and repeatable testing. We use NVIDIA Isaac Sim, a robotics simulation platform built on the PhysX physics engine, as our simulated environment. Isaac Sim provides realistic physics, including rigid body dynamics and frictional contact, which are crucial for accurately modeling a skid-steer robot's behavior. The key aspects of the environment setup are as follows:

The robot considered is a four-wheeled skid-steer vehicle with independent wheel drive. All four wheels

Table 4.1: Vehicle dynamics parameters

Description	**Symbol**	**Value**
Vehicle mass	m	25 kg
Track width	B	0.5 m
Distance from front axle to CG	L_f	0.2075 m
Distance from rear axle to CG	L_r	0.2075 m
Wheel radius	R	0.1175 m
Vehicle inertia around CG	J	1.05 kg·m^2
Friction coefficient	μ_f	0.85
Rotation moment of wheel	j	0.85 kg·m^2
Max torque of wheel	T_{max}	3 N·m
Min torque of wheel	T_{min}	0 N·m

are identical and are mounted rigidly to the chassis (no active steering joints). The vehicle's physical parameters, such as mass and inertia, can be found in the Table 4.1. Each wheel has a certain radius and mass; however, rather than simulate suspension, we assume cylindrical rigid bodies for each wheel. The actuation is torque control at each wheel. We impose torque limits $[T_{\min}, T_{\max}]$ on the action outputs to reflect motor limitations – for instance, each wheel motor may be limited to 3Nm of torque (the exact value depends on the specific robot's actuators). If the agent's policy outputs a torque beyond this range, it is clipped to the nearest limit. These bounds not only keep the simulation stable but also teach the agent to operate within physically feasible regimes, This can be seen as another form of expert guidance by shrinking the action space in order to make exploration easier. In our simulations, the robot operates on a flat horizontal surface with a uniform friction coefficient (e.g. $\mu = 0.85$, representative of dry asphalt). This moderate friction value allows some wheel slip during aggressive maneuvers, mimicking real skid-steering behavior. The physics integrator updates the vehicle's pose (position and orientation) and velocities at a frequency of 600 Hz in order to allow the simulator to calculate realistic friction interactions for the current forces/torques. We chose a fixed control frequency of 10Hz for the agent, meaning the agent outputs a torque action every 0.1s. The simulation internal physics step can be smaller for accuracy of skid physics, but the agent's decision and observation cycle is at 10Hz. This is fast enough for the dynamics of typical straight driving with skid-steer vehicles. However this assumption breaks when the skid-steer vehicle accomplishes rapid locomotion and enters a high-slip regime. Unfortunately, this 0.1s is necessary to allow the consequences of the actions to develop, expressive enough for a strong learning signal to develop. If the actions are given a much shorter timestep to evoke a change in the state the observed reward might be to small to create a learning signal.

Table 4.2: Hyperparameters of the KA-DDPG algorithm

Parameters	**Value**
Random seed	20
Max episode	5000
Max steps per episode	100
Step size	0.1
Memory capacity	1,000,000
Batch size	512
Actor network learning rate	0.0003
Critic network learning rate	0.001
Exploration noise scale	0.1
Soft update rate	0.01
Discount factor	0.999
Proportional coefficient	0.035

4.5.1. Hyperparameters

All experiments are run on an NVIDIA's A40 GPU using Delft Artificial Intelligence Cluster. The training framework (DDPG and KA-DDPG algorithms) is built using Python and PyTorch (for neural network function approximation). The actor and critic networks in our agent are both feed-forward neural networks with three hidden layers (ELU activations) of sizes 512, 512, and of 128 neurons respectively per layer, which is sufficient given the low-dimensional state and action spaces. The output layer of the actor uses a sigmoid activation to limit actions between 0 and 1. These actions are scaled to the range of allowed torques, ensuring the actions fed to the environment are within bounds. Figure 2.3 showed the network structure. Notice how the actions of the policy network are used as observations by the critic network. This illustrates the off-policy character that allows for easy action injection. Standard DDPG hyperparameters are employed (replay buffer size, discount factor $\gamma = 0.99$, soft target update coefficient τ, etc.), as summarized in the Table 4.2 below. These settings are kept constant across all experiments for consistency.

4.6. Summary

In summary, our methodology now consists of Knowledge-Assisted Mixed Mode Actioning (KAMMA) combined with a curriculum schedule. In the next chapter, we will evaluate this approach experimentally. We will compare variants including: baseline RL with various action spaces, the original KA-DDPG (blending) method, the KAMMA method, and finally the KAMMA method with each of the two curricula. This will allow us to quantify the contributions of both KAMMA and curriculum learning to the overall performance.

5

Experiments and Results

5.1. Evaluation Protocol

The primary goal of the experimental evaluation is to assess the quality and consistency of trajectory-following behavior learned by different reinforcement learning agents. Given the challenges posed by non-holonomic constraints, dynamic coupling, and slip-dominated terrain, traditional performance metrics such as episodic return or an arbitrary success rate are insufficient to characterize policy effectiveness. We therefore introduce two complementary metrics: tracking error and torque smoothness.

Tracking Error To evaluate the accuracy of the learned policies, we define tracking error as the absolute deviation between the commanded velocity $v_{\text{cmd}}(t)$ and the actual velocity $v_{\text{act}}(t)$ observed during an evaluation trajectory:

$$\epsilon(t) = |v_{\text{cmd}}(t) - v_{\text{act}}(t)|. \tag{5.1}$$

This metric captures the agent's ability to accurately follow a target velocity profile, which is critical for safe and predictable control. Evaluation runs are conducted using an every episode increasing $v_{\text{cmd}}(t)$. We increase the $v_{\text{cmd}}(t)$ for 50 episodes from 1.0 m/s to 2.0 m/s, to test policy generalization across a range of operating regimes. Each agent is evaluated over five random seeds to assess consistency.

Smoothness (Jerk) Beyond tracking accuracy, we examine the internal control behavior using a smoothness metric related to jerk (change in control inputs). Specifically, we measure the mean squared difference between consecutive torque commands for each wheel:

$$S = \frac{1}{T-1} \sum_{t=1}^{T-1} \|\tau_{t+1} - \tau_t\|^2. \tag{5.2}$$

where τ_t is the vector of wheel torques at time step t. This metric reflects the temporal regularity of the control signal: high-frequency oscillations or erratic torque changes result in large jerk (high smoothness values), indicating poor smoothness. In the context of torque control, a lower smoothness value (less jerk) signifies more coherent, stable behavior. We also use smoothness as a proxy for policy convergence, a well-converged, high-performing agent is expected to exhibit low tracking error and stable jerk. As training progresses, effective policies should evolve toward smoother torque changes that still achieve the desired velocities. In this sense, smoothness provides insight into the maturity and reliability of the learned control strategy, complementing the tracking error metric.

5.1.1. Protocol Overview

In addition to tracking error and torque smoothness—measured over a velocity ramp, we also report the convergence episode to capture training efficiency, and assign a qualitative stability score (High, Medium, Low) for each policy. This score reflects internal reliability based on: (1) convergence behavior across seeds, including visual inspection of reward curves for divergence or oscillation; (2) post-training

torque smoothness as a proxy for physically coherent control and avoidance of erratic behavior; and (3) robustness over dynamic changes, where sharp drops in tracking performance, particularly at the working point of the controller (e.g., ± 1.5 m/s), signal instability or catastrophic forgetting. While these indicators do not substitute for formal robustness tests, they allow a structured heuristic comparison. We retain the term “stability” to emphasize consistent behavior across conditions, though it partially overlaps with robustness in this setting. This evaluation framework is applied consistently in all experiments throughout Sections 5.2–5.4.

5.2. Baseline: KA-DDPG Variants

This section presents the baseline results for the Knowledge-Assisted DDPG (KA-DDPG) approach across varying action space dimensionalities: 1D (single combined torque), 2D (left/right differential torque), and 4D (individual wheel torques). We evaluate both the training behavior (torque outputs during training) and the final policy quality (trajectory smoothness and velocity tracking on the test ramp) for each variant. The expectation is that reducing the action dimension simplifies the learning problem (leading to faster convergence and smoother control), whereas higher-dimensional action spaces offer more expressive control but are harder to train due to increased complexity.

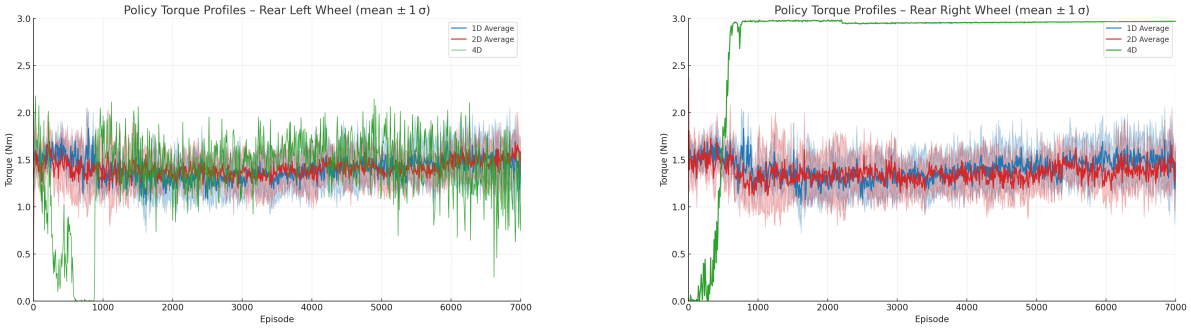


Figure 5.1: Rear Left and Rear Right torque profiles during training for 1D (blue), 2D (red), and 4D (green) KA-DDPG variants (five seeds each).

5.2.1. Training-Run Torque Profiles

Figure 5.1 shows the evolution of wheel torque commands during training for the three KA-DDPG variants. As expected, the 1D and 2D policies learn a stable torque output very quickly. The 1D agent’s torque profile is smooth and settles early in training, confirming the expectation that a low-dimensional action space yields a simpler optimization landscape and rapid convergence. With only one control parameter, the 1D agent cannot induce wheel-specific oscillations, which effectively reduces exploration noise. This simplicity comes at the cost of expressiveness, but it clearly leads to faster stabilization – the agent finds a single torque signal that successfully drives the vehicle with minimal tuning.

In contrast, the 4D variant (independent wheel torques) was expected to face difficulty due to its high-dimensional action space, and the observed training behavior confirms this. The 4D agent fails to produce coherent or convergent torque patterns: its outputs saturate and jitter wildly throughout training. This unstable behavior indicates destructive interference between the four torque outputs – essentially, the agent struggles with credit assignment when each wheel is controlled separately. This is recognized in the torque profiles by having one wheel (Rear Left in Figure 5.1a) capable of stable torques around 1.5 Nm, while the other three wheels of which Rear Right is taken as representative in Figure 5.1b saturates and never recovers. This is also illustrative for the results of dynamic coupling. Results for all wheel torque profiles will be available in the Appendix. The result aligns with our expectation that unrestricted 4D control is hard to learn: exploration noise in each wheel channel and conflicting gradients from the critic prevent the policy from finding a stable solution. The continuous blending of actions in this high-dimensional space muddles the feedback signal, so the agent receives mixed and ambiguous reward contributions from each wheel’s torque. As predicted, the 4D variant does not converge within a reasonable training time, highlighting severe training challenges when every wheel is adjusted independently.

The 2D variant (differential torque control) represents an intermediate case. Initially, its torque outputs are more chaotic than the 1D's (reflecting the added degrees of freedom), but over time the 2D agent manages to converge to a symmetric torque profile between the left and right wheel pairs. This suggests the 2D policy learns to coordinate the two sides: torque commands for left and right wheels stabilize into a pattern that is roughly mirror-symmetric, with structured temporal variations. We expected the 2D agent to be more stable than 4D but perhaps slower to converge than 1D. Indeed, the observed behavior confirms an intermediate outcome – the 2D agent does converge, albeit more gradually than 1D, and ends up producing reasonable, interpretable torque trajectories. This pattern supports a fundamental trade-off: finer control granularity (as in 4D) offers more expressive power in principle, but it greatly complicates learning, whereas a moderate dimensionality like 2D strikes a practical balance between expressiveness and trainability. In summary, the training-run results validate our expectations: simpler action spaces converge faster and more smoothly, while overly complex action spaces (4D) can fail to converge at all.

5.2.2. Evaluation-Run Smoothness and Tracking Error

Figures 5.2 and 5.3 present the evaluation results for each baseline policy in terms of trajectory smoothness and velocity tracking error, respectively, over the stepped velocity ramp. Each policy was trained five times; the shaded regions in the plots indicate the variability across seeds, providing a sense of robustness.

Smoothness (Jerk) – 5.2: We initially expected the 1D policy to yield the smoothest control (lowest jerk) because a single, shared torque command should discourage wheel-to-wheel mismatches. At low speeds this expectation is met: the 1D agent shows very low jerk and highly stable outputs. Yet at higher speeds the data reveal a different picture. Two complementary factors now dominate.

(1) Velocity-noise sensitivity. Because the 1D policy can correct the robot's longitudinal velocity only by raising or lowering the common torque level, it has fewer degrees of freedom to attenuate small sensor or estimation errors in forward speed. When those errors grow with velocity, the 1D agent must over-compensate, producing the burst of oscillatory torques we observe as a jerk spike in one of the high-speed seeds.

(2) Expert-alignment advantage. The expert demonstrator inherently issues differential (2D) torque commands, so the 2D policy's action space is perfectly aligned with the expert's strategy. That alignment lets the 2D agent harvest more reliable gradients during imitation learning, giving it a robust initialization that persists into the high-speed regime.

In combination, these effects explain why the 2D variant ends up with consistently lower jerk and tighter inter-seed variance across the full velocity ramp, while the 1D policy, despite quick convergence and excellent smoothness at low speeds, occasionally destabilizes when velocity noise rises. As expected, the 4D variant is omitted because it never produced a deployable policy, so its smoothness curve would

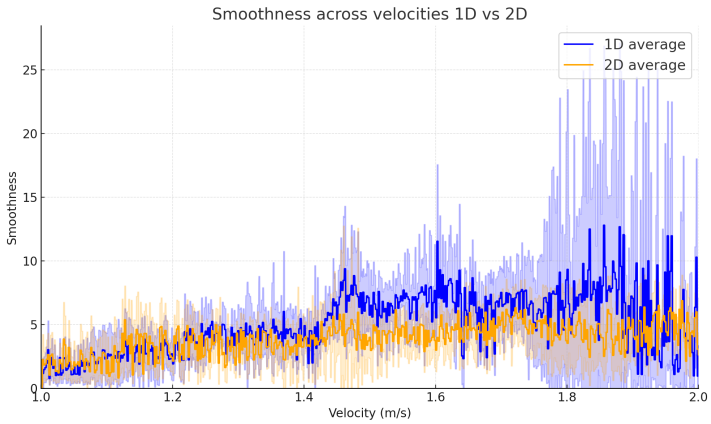


Figure 5.2: Smoothness over velocity ramp for 1D and 2D KA-DDPG (five seeds each), with the average over five seeds.

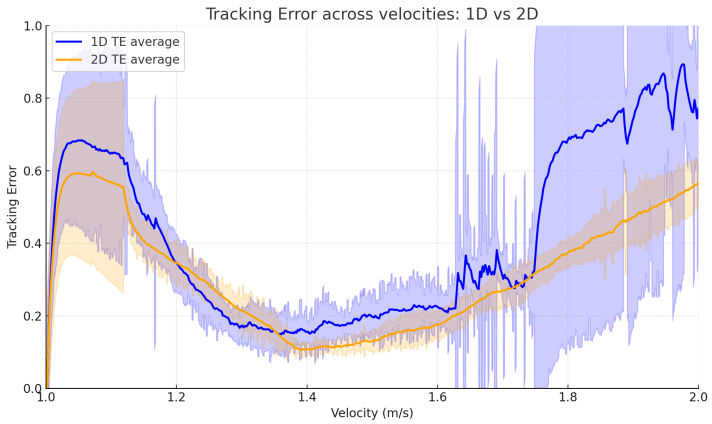


Figure 5.3: Tracking error over velocity ramp for 1D and 2D KA-DDPG (five seeds each), with the average over five seeds.

not be meaningful. Overall, the 2D agent achieves the best smoothness (lowest jerk with tight seed clustering) and superior high-speed stability, underscoring the twin importance of noise-robust action flexibility and compatibility with the expert's control signals.

Tracking Error – Figure 5.3: We anticipated that all agents would track best around the Controller's nominal operating speed ($\pm$1.5 m/s) and that tracking error would worsen (rise) at very low and very high speeds where the task is more challenging. The results confirm this general trend for both 1D and 2D policies. As shown in Figure 5.3, each curve exhibits a clear dip in error around 1.4–1.5 m/s (the expert's "working point") and a steeper increase in error as the commanded velocity approaches the extremes. Both agents thus met the expectation of good accuracy in the middle of the range and decreasing accuracy toward the boundaries. Between the two, the 1D agent achieves low average error with tight error bounds in the low-speed region, performing about as well as the 2D agent in that regime. However, as the velocity increases beyond $\pm$1.5 m/s, the 1D policy's error grows much more rapidly than the 2D's. Figure 5.3 shows that at the top of the velocity ramp ($\pm$2 m/s), the 2D agent's tracking error remains about half that of the 1D agent, confirming superior high-speed control. It also confirms our expectation that the 1D agent's limited action expressiveness and its slight misalignment with the expert's multi-wheel control would hurt its high-speed tracking. In fact, the 1D tracker's error becomes unacceptably large at the highest speeds, whereas the 2D agent maintains significantly lower error in these high-speed conditions. The improved high-velocity tracking of the 2D policy is likely due to its ability to differentially adjust left vs. right torque, giving it more control authority to counter slip and maintain traction as speed increases. As before, the 4D agent's tracking error is not plotted because that variant never converged properly.

5.2.3. Lessons on Action Dimensionality

The results in Table 5.1 confirm that using a lower-dimensional action space greatly facilitated learning for skid-steer control. The 1D action variant converged quickly (reaching its best performance by 2000 episodes) and excelled at low-speed tracking, but its limited actuation flexibility led to large errors at high speeds (qualitative stability rated "Medium"). In contrast, the 2D action agent achieved a significantly lower final tracking error (0.321 vs 0.425) and smoother control inputs (smoothness 3.89 vs 5.10) than the 1D agent. The 2D policy maintained stability even during a challenging velocity-ramp test (stability "High"), likely because differential wheel torque control helped counteract slippage at higher velocities. Meanwhile, a fully expressive 4D action space proved too difficult to train (failing to converge), reinforcing the need to avoid overly complex action dimensions. In summary, constraining the action space to an appropriate dimensionality can streamline learning and improve high-speed robustness, so long as it remains expressive enough to handle the vehicle's operational range.

Variant	Tracking Error (±)	Smoothness (±)	Conv. Ep.	Stability
1D	0.425 ± 0.374	5.10 ± 4.89	2000	Medium
2D	0.321 ± 0.185	3.89 ± 2.30	2000	High

Table 5.1: Tracking Error and Smoothness (± std), convergence episode, and qualitative stability for Section 5.2 (1D vs 2D).

5.3. KAMMA vs. KAMMA IL

Having established baseline behavior, we next evaluate our proposed Knowledge-Assisted Mixed Mode Actioning (KAMMA) strategy and compare it against an ablated version and the original expert controller. In this context, KAMMA refers to the agent that uses mixed guiding/observed rewards, and expert/agent actions with a decaying probability of expert intervention (as described in Section 4.3), and KAMMA IL ablation refers to a knowledge-assisted agent that uses only imitation learning (expert guidance) without any subsequent reinforcement learning phase on the observed reward. In addition, we include the expert's own performance (the baseline Controller) for reference. The experiments here focus on the 4D action setting (individual wheel torques) because KAMMA was designed to address the challenges observed with the 4D baseline. We analyze training stability via torque profiles and then assess the smoothness and tracking performance after training. Our expectation was that KAMMA's discrete switching approach would avoid the instability seen in the 4D blending method, yielding more stable training, and that after the expert is phased out, the RL component would allow the agent to surpass the KAMMA IL agent's performance, especially in regimes where the expert was weak.

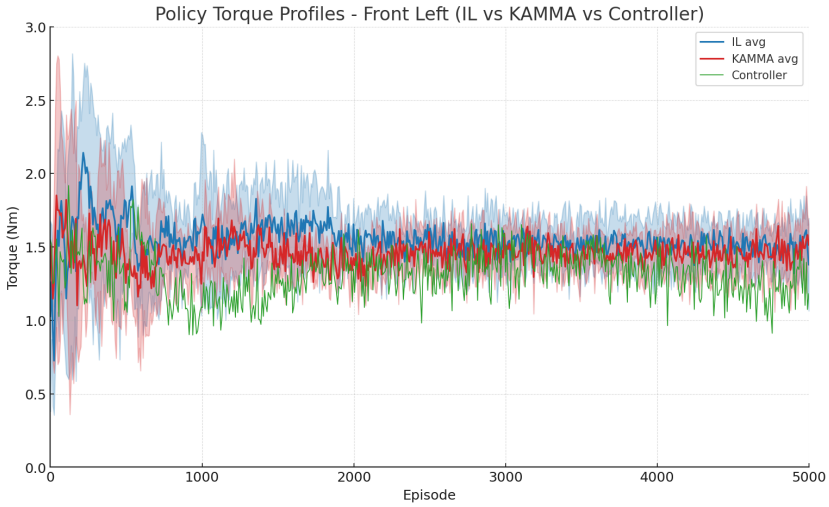


Figure 5.4: Front Left training-run torque profiles for single baseline Controller seed, KAMMA, and IL KAMMA ablation (five seeds each).

5.3.1. Training-Run Torque Profiles

Figure 5.4 represents the torque output during training for a representative run of the baseline Controller (expert) and the average behavior of the KAMMA and KAMMA IL agents (with five seeds each). As expected, the expert Controller's torque commands (red curve) remain very stable and consistent throughout training – the expert is a hand-crafted or previously tuned policy, so its control signals exhibit steady amplitude and minimal noise by design. The KAMMA agent (blue curve) and the KAMMA IL ablation (green curve) show similarly stable torque trajectories during training. This immediate stability confirms our expectation that KAMMA's mixed-mode strategy would preserve the early training reliability provided by the expert. In fact, we observe that both KAMMA and KAMMA IL maintain tightly grouped torque profiles across seeds, indicating that the learning process is well-behaved and produces consistent outputs (no wild oscillations) from the start. The absence of erratic torques stands in stark contrast to the earlier 4D blending case (Section 5.2.1), demonstrating that discrete expert-agent switching indeed leads to more interpretable and stable training behavior. By executing either the full expert action or the full learned action at each step (instead of a blend), KAMMA ensures that the agent's updates are based on clear, attributable decisions. This allows the policy to refine its torque

outputs steadily over time without the interference issues seen in the continuous-blending approach. We even observe a similar to even earlier emergence of sensible control structure in KAMMA's torque traces than we saw in the "easier" low-dimensional KA-DDPG (1D/2D) baselines.

Interestingly, the KAMMA IL agent displays surprisingly competent torque patterns very early in training. Essentially, even without any reward from environment interaction, the KAMMA IL agent is leveraging the guiding reward (which imitates the expert's action) to mimic the expert policy to a significant degree. This was expected to some extent (since the KAMMA IL agent is designed to copy the expert while conditions do not change), but the level of proficiency right from the start highlights how effective the guiding reward is. Using the guide as a surrogate training signal will be a strong correction in later episodes and realize robust exploration. The KAMMA IL agent's torque outputs closely follow the expert's behavior in the initial stages, confirming that imitation alone can produce a reasonable policy within the expert's performance envelope. However, we expect the KAMMA IL agent's progress to plateau over time. Because it lacks any observed reward feedback, the IL agent has no incentive to further improve beyond what it has learned from the expert. In theory, it should asymptotically approach the expert's policy and then stagnate. We anticipate that without reinforcement learning, the KAMMA IL agent will not learn to correct the expert's shortcomings in unfamiliar regimes (e.g. low speed, high slip, or high speed) and will remain limited by the expert's suboptimal behaviors. This expectation will be examined by comparing the IL agent's tracking performance to the full KAMMA agent's performance: if KAMMA does not show clear improvements over KAMMA IL in those regimes, it would indicate that the RL phase failed to refine the policy further (implying we might need to adjust the reward design or training schedule). In summary, the training traces suggest that both KAMMA and the KAMMA IL agent achieve stable learning thanks to expert guidance, and KAMMA's approach avoids the chaotic torque outputs seen in the 4D baseline. The key question remains whether KAMMA's additional RL phase yields measurable gains over pure imitation; a purely IL agent should eventually stall (plateau) at the expert's level, never exceeding it, since it only learns "what the expert would do" but not why or how to do better.

5.3.2. Evaluation-Run Smoothness and Tracking Error

After training, we evaluated the KAMMA and KAMMA IL policies on the standardized velocity ramp, measuring smoothness and tracking error to compare their performance and convergence quality. The expectation was that KAMMA, by incorporating an RL phase, would achieve equal or better smoothness than the KAMMA IL agent and better tracking accuracy especially outside the expert's comfort zone (low and high speeds). Figures 5.5 and 5.6 summarize the results alongside the baseline curves for reference.

Smoothness – Figure 5.5: The KAMMA agent consistently achieves low jerk across the velocity ramp, staying below a smoothness value of about 10 for all tested speeds. This level of smoothness is a dramatic improvement over the unstable 4D KA-DDPG baseline (from Section 5.2) which showed

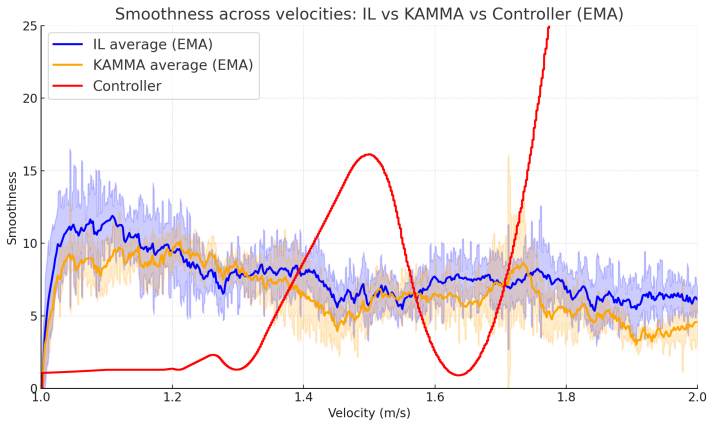


Figure 5.5: Smoothness over velocity ramp for KAMMA and KAMMA ablation (five seeds each), with the average over five seeds and the expert Controller.

erratic, high-jerk behavior, and it is only slightly worse than the best-case smoothness we observed in the 2D action-space variant. This outcome is in line with our expectations: by the end of training, the KAMMA policy produces nearly as smooth control as the simpler 2D policy, despite operating in the more challenging 4D action space. The small increase in jerk relative to 2D is expected because having four independent torque outputs allows for some wheel-to-wheel differences (greater expressivity) and thus can introduce minor high-frequency components. Importantly, the KAMMA smoothness curves for all five seeds are tightly clustered and low-valued, indicating a high level of consistency and stability in the learned behavior. In effect, KAMMA's smoothness performance confirms that its policy not only converged but did so without sacrificing control quality.

Perhaps the most revealing aspect is the comparison between KAMMA and the KAMMA IL agent. We anticipated that KAMMA's RL phase might further smooth out the policy or at least not degrade smoothness relative to the KAMMA IL agent. The results show that both KAMMA and KAMMA IL achieve almost identical smoothness profiles over the entire velocity range. Both agents maintain low jerk even at the highest speeds, which is encouraging, but it was unexpected to see such similarity. We had presumed that without additional RL fine-tuning, the KAMMA IL agent might either have higher jerk at the extremes or conversely that KAMMA might introduce a bit more jerk while searching for higher performance. Neither happened: instead, the KAMMA IL policy is almost as smooth as KAMMA's. This suggests that the imitation process alone already yielded a very smooth policy constrained by the expert's inherently smooth behavior at the working point and the formulation of the guiding rewarde. It also raises the possibility that KAMMA's RL phase did not significantly improve (or worsen) smoothness, perhaps because the observed reward weighting and curriculum were not tuned to explicitly optimize smoothness further. In other words, the RL component focused more on reducing tracking error and may have left the "smooth enough" policy largely unchanged in terms of jerk.

Another interpretation is that both KAMMA and KAMMA IL share the same mechanism (the guiding reward and action suppression early on) that brings the policy to a certain smooth operating point benefiting from the 2D controller. Improving smoothness in the 4D action space would inherently create more individual jerk which makes deviating with the RL agent much harder. This outcome, while positive in that smoothness remained low, is a bit paradoxical: we assumed smoothness might degrade at high velocities without careful reward shaping, yet both agents remained smooth even when actual velocity reached ranges where the expert itself tends to be less smooth. This indicates there may be subtle effects of the guiding reward that inherently promote smooth control (for example, the guiding reward penalizes wheel torque differences on a per-wheel basis, which could impose smoothness indirectly). Regardless, the key takeaway is that KAMMA did not compromise smoothness: it managed to keep jerk low, on par with pure IL, thus meeting the expectation of stable control quality.

Tracking Error – Figure 5.6: In terms of tracking accuracy, we expected KAMMA to outperform both the KAMMA IL agent and the simpler baseline agents (1D/2D) at the edges of the performance

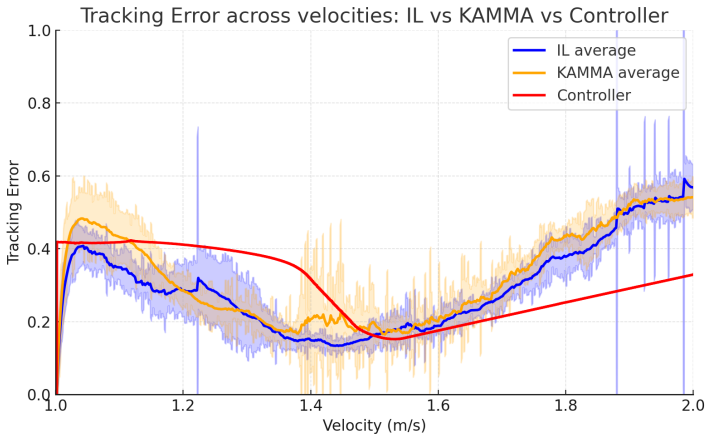


Figure 5.6: Tracking error over velocity ramp for single baseline Controller seed, KAMMA, and KAMMA ablation (five seeds each), with the average over five seeds and the expert Controller.

envelope (where the expert data was scarce or less reliable). The results support this expectation in several ways. First, KAMMA achieves lower tracking error than the baseline KA-DDPG agents across most of the ramp. In particular, compared to the 1D and 2D results from Section 5.2, the KAMMA agent's error curve stays lower in both the low-speed and high-speed regions. This demonstrates that the expressiveness of the 4D action space can pay off when properly trained: with KAMMA, the 4D agent finally outperforms the constrained 1D/2D agents in terms of tracking error, validating the idea that our method unlocks the potential of the higher-dimensional control. Second, focusing on the KAMMA vs. KAMMA IL comparison, we see that the KAMMA IL agent actually has an edge around the Controller's working point ($\pm$1.4 m/s), whereas KAMMA shows better performance at the lower and higher extremes. Specifically, near 1.4 m/s (the middle of the ramp), the KAMMA IL agent's error dips slightly below KAMMA's error. This makes sense: around that speed, the KAMMA IL agent is essentially doing exactly what the expert would do, and since that is precisely the regime the expert is tuned for, the IL agent excels there. KAMMA at the same mid-range speed performs nearly as well, but the small discrepancy suggests that its additional exploration might not have improved performance in the regime the expert was already good at, a minor trade-off for having more freedom elsewhere. At the high-velocity end ($\pm$2 m/s) and low-velocity start ($\pm$1 m/s) of the ramp, however, KAMMA sporadically achieves lower error than the KAMMA IL agent. This indicates that in those challenging regimes (high slip in fast motions, or difficulties in very slow precise control), the RL phase tried to help KAMMA refine the policy beyond what imitation alone provided. We do observe that KAMMA's advantage at the extremes comes with higher variance: some KAMMA runs greatly outperform the IL agent at high speed, while others are on par or slightly worse, leading to larger error spread at the far right of the plot. This variability was not entirely unexpected. Once the expert's influence fades, the KAMMA agents rely on pure RL in a tough part of the state space, which can result in instability for some seeds. Indeed, the increasing variance of KAMMA's error at high velocities points to occasional instability, echoing the earlier concern that the agent might struggle once fully on its own in unseen conditions. Nonetheless, the trend aligns with our hypothesis: the KAMMA IL agent cannot improve beyond the expert's capabilities, whereas KAMMA, through reinforcement learning, is able to push the performance envelope (even if not uniformly across all seeds). KAMMA's ability to sometimes control the vehicle more accurately in high-speed scenarios where the expert (and thus the IL agent) likely would struggle, provides evidence that the RL phase is doing its job in principle, even if there is room to make it more consistent.

5.3.3. Lessons on Action Mixing

Table 5.2 compares the proposed KAMMA agent with its KAMMA IL ablation and the expert controller. The KAMMA imitation agent (KAMMA IL) quickly achieved expert-level performance, its final tracking error (0.300) essentially matched the hand-crafted controller (0.299), and it produced very smooth actions, inheriting the expert's stability in nominal conditions. However, as expected, the KAMMA IL policy could not improve beyond the expert's capabilities, struggling in regimes outside the expert's experience (e.g. high and low speeds). By incorporating an RL phase, the full KAMMA agent maintained comparably low error (0.320) while further reducing the smoothness metric (6.85 vs 8.60, indicating smoother control than IL). More importantly, KAMMA expanded the operational envelope: it remained stable under a demanding velocity ramp (stability "High") and could sometimes control the vehicle at extreme speeds where the expert controller generated very large smoothness values. The expert controller tracks well in its intended operating range but falls apart at higher speeds, it was not designed for the extremes of our task. Indeed, we observed the controller often saturating and losing stability beyond $\pm$ 1.5 m/s, hence its stability is rated Low in Table 5.2. In short, mixing expert guidance with autonomous learning preserved early stability and smoothness while allowing the agent to surpass the expert-based policy in challenging conditions, confirming the benefit of an RL fine-tuning stage after

Variant	Tracking Error ($\pm$)	Smoothness ($\pm$)	Conv. Ep.	Stability
KAMMA IL	0.300 $\pm$ 0.136	8.60 $\pm$ 6.80	2000	High
KAMMA	0.320 $\pm$ 0.143	6.85 $\pm$ 2.62	2000	High
Controller	0.299 $\pm$ 0.093	13.11 $\pm$ 14.75	-	Low

Table 5.2: Tracking Error and Smoothness ($\pm$ std), convergence episode, and qualitative stability for Section 5.3 (KAMMA vs IL vs Controller).

imitation learning.

5.4. Curriculum-Enhanced Variants

Having established that KAMMA enables stable policy learning in the difficult 4D torque control setting, we next investigate whether curriculum learning can further enhance training outcomes. The idea is to shape the learning process by staging the difficulty of the task, potentially yielding better convergence or final performance than a one-phase training. We evaluate two curriculum strategies, Growing Variance (GV) and Bimodal Drift (BD), each applied to the KAMMA agent. In the GV curriculum, the distribution of target velocities starts narrow (low variance around a base speed) and grows gradually, exposing the agent to higher speeds and more slip only as it masters slower speeds. In the BD curriculum, easy (low-speed) and hard (high-speed) tracking tasks alternate in a bimodal sampling scheme, forcing the agent to continuously switch contexts (see Section 4.4). We test each curriculum both in isolation and in combination with a FIFO replay buffer, which is a modified replay memory that always retains the most recent experiences (discarding older ones) to emphasize recent training phases. This allows us to study not only the effect of the curricula themselves but also the influence of a curriculum-sensitive memory mechanism in prioritizing recent interactions. The expectation is that a well-designed curriculum will improve training stability and final performance (especially at high speeds) compared to the non-curriculum KAMMA, and that the FIFO replay buffer will help enhance convergence speed while risking catastrophic forgetting of proficiency on already learned sub-tasks.

5.4.1. Training-Run Torque Profiles

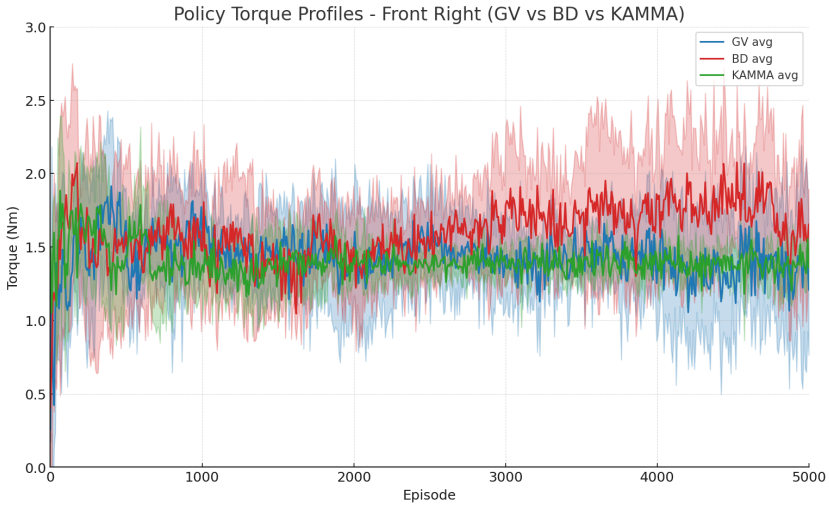


Figure 5.7: Front Right training-run torque profiles for Growing Variance (GV), and Bimodal Drift (BD) compared to the KAMMA base

During the initial IL-guided phase, both GV and BD agents converge to stable torque behaviors, as expected. The expert's guidance ensures that early training (moderate speeds for GV, alternating slower/faster for BD) yields coherent torque outputs. We anticipated that throughout training the GV strategy might maintain more stable behavior than BD, due to its gentle difficulty progression. The observations partly support this: both GV and BD show a similar overall trend initially, converging to reasonable torque patterns under IL. However, as training progresses and the agent is exposed to higher speeds (when the RL component starts dominating), all variants eventually begin to diverge again toward the end of training. In Figure 5.7, we see that the torque profiles for both GV and BD converge early (during the IL-heavy phase) but then diverge later in training, exhibiting growing variance and oscillations in the final part of the learning curve. This late divergence was not fully expected. Ideally, a curriculum would lead to sustained stability, but it indicates that once the agent is on its own (relying mostly on RL in harder tasks), the challenges re-emerge. The GV curriculum's profile appears more stable than BD's overall, but notably both curricula show this end-of-training uptick in instability through high variance.

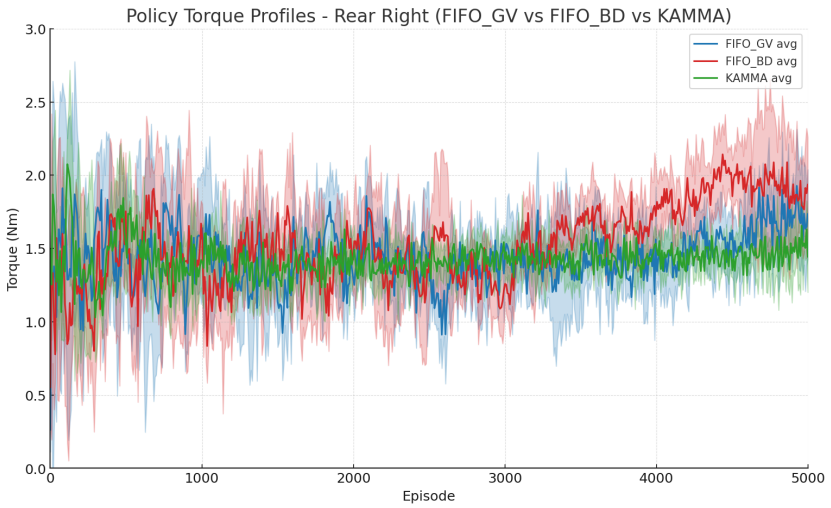


Figure 5.8: Front Right training-run torque profiles for the FIFO Replay versions of Growing Variance (GV), and Bimodal Drift (BD) compared to KAMMA base

Comparing GV vs. BD, we do observe that the BD curriculum induces more erratic torque fluctuations during training than GV does. Especially in the latter half of training, the BD agent's torque trace is quite jagged and variable, whereas the GV agent's torque, though diverging, is a bit more steady in amplitude. This matches our expectation that frequent context-switching (BD) would inject additional instability. The alternating easy/hard regime in BD exposes the agent to a broader range of states early on, which is intended to improve generalization, but it also seems to hamper convergence: the agent has to constantly re-adjust between slow and fast trajectories, which likely disrupts the learning momentum. Without a proper mechanism to retain what was learned in one context while jumping to another, the BD agent experiences something akin to task-switching instability.

Additionally it can be concluded that the IL phase did not sufficiently impart a general understanding of the system's dynamics, the agent likely didn't fully learn the vehicle's slip behavior using only imitation. Gradual exposure to higher slip, when paired with a fast decay of IL influence, proved insufficient to maintain stability later on. This system identification method was always implicit but seems to be more effective using the observed reward as this better relates the complex system dynamics like velocity and slip. Thus we have to conclude that these curricula sampling methods need a different, probably slower, decay schedule to gain more exposure to the observed reward. This will create more opportunity to instill that system understanding. The curriculum sampling methods did extend the agent's operational range, gradual exposure to both low and high speeds appeared to improve generalization to extreme conditions, as the curriculum-trained agents handled high speeds better than the base KAMMA agent. However, this benefit came with the caveat that performance at some intermediate regimes suffered (indicating some forgetting or lack of retention of mid-speed proficiency). The early IL phase might already create value anchors that are valuable during the later RL updates.

Now, looking at the FIFO replay buffer effect (Figure 5.8), the expectation was that FIFO would exacerbate some of the forgetting by always reinforcing recent tasks, but also cause faster convergence due to a smaller and more focused data distribution. Consistent with this expectation, both FIFO-enhanced variants (GV+FIFO and BD+FIFO) exhibit faster but less stable initial convergence than their standard counterparts. In Figure 5.8, the torque profiles for GV+FIFO and BD+FIFO settle into a pattern more quickly, but early-training variability is high compared to Figure 5.7. This suggests that prioritizing recent experiences helps the agent focus on the current task difficulty and solidify those skills without being constantly distracted by replay of old experiences. The FIFO variants also show a divergence in torque outputs towards the end of training as expected. Interestingly, the oscillations are somewhat less extreme than without FIFO). This outcome is a partial surprise: we expected the FIFO buffer to exacerbate the late-training deterioration, but the results indicate it postpones or reduces it. The likely reason is that while FIFO inherently discards older data that might still be valuable for maintaining performance on earlier phases, it also emphasizes recent data which is good for the current phase and

final performance. The agent might be losing some of the “lessons” from earlier speeds once those samples are dropped from the buffer, thus a mild form of forgetting sets in – just later and less severely. Another factor is that model capacity and training dynamics can also cause forgetting: as the distribution of tasks shifts, the neural network may reallocate its capacity to new situations at the expense of older ones, especially if the buffer is very limited in size. In summary, FIFO improves speed of convergence as expected, and it has a nuanced effect on stability and performance. Notably, even with FIFO, BD’s torque profile remains more erratic than GV’s. The BD+FIFO agent benefits from recency bias (less forgetting than BD alone), yet the inherent difficulty of switching between disparate tasks still leads to visible torque oscillations (albeit somewhat muted). This reinforces the idea that curriculum design (task order) plays a larger role in stability than the replay scheme: a chaotic curriculum like BD yields chaotic training traces, while a memory tweak like FIFO can partly compensate for that.

5.4.2. Evaluation-Run Smoothness and Tracking Error

We now evaluate the final performance of the four curriculum-enhanced variants on the test velocity ramp, focusing on smoothness (jerk) and tracking error metrics (Figures 5.9 and 5.10 respectively). For ease of comparison, we discuss trends relative to what we observed with non-curriculum KAMMA. Our expectations were that the GV curriculum would yield the smoothest, most accurate policy (due to its structured learning progression), that the BD curriculum might show more variability (due to its oscillating nature), and that the FIFO replay could destabilize these metrics by causing catastrophic forgetting.

Smoothness – Figure 5.9: The GV curricula indeed produces the most stable (lowest jerk) policies overall, which confirms our expectation about the benefit of a gradual curriculum. Across the combined ten seeds, the GV agents achieve consistently low average smoothness values at most speeds. The progressive exposure to higher speeds appears to have allowed the agent to refine its control incrementally, avoiding sudden jumps in policy that cause jerk. By contrast, the BD curricula results in slightly higher jerk levels. The BD curves in Figure 5.9 show noticeably larger smoothness values. This aligns with our prediction that the alternating task difficulty in BD would introduce control inconsistency. The BD agents’ policies dosn’t settle as smoothly, likely because it never fully “finishes” learning one regime before switching to another. In other words, the BD-trained policy is less smooth due to the stop-and-go learning process.

It’s worth noting that the GV smoothness results, while low on average, are not perfectly flat. The GV agent’s jerk is generally low but shows some variation across the velocity range. In fact, somewhat unexpectedly, the GV agent exhibits a few sporadic increases in smoothness at certain speeds (including a slight uptick even around the mid-speed range of 1.4 m/s, where one would expect performance to be best). These minor irregularities in the GV smoothness plot hint that even a gradual curriculum does not entirely eliminate challenges; the agent might still experience some difficulty at transition points in the curriculum (e.g., when moving from one stage to the next, there could be a brief rise in jerk). Nonetheless, the key improvement with GV is seen at the low-velocity end: the GV-trained agents maintain relatively low jerk even as speed approaches 1.0 m/s, whereas the non-curriculum KAMMA

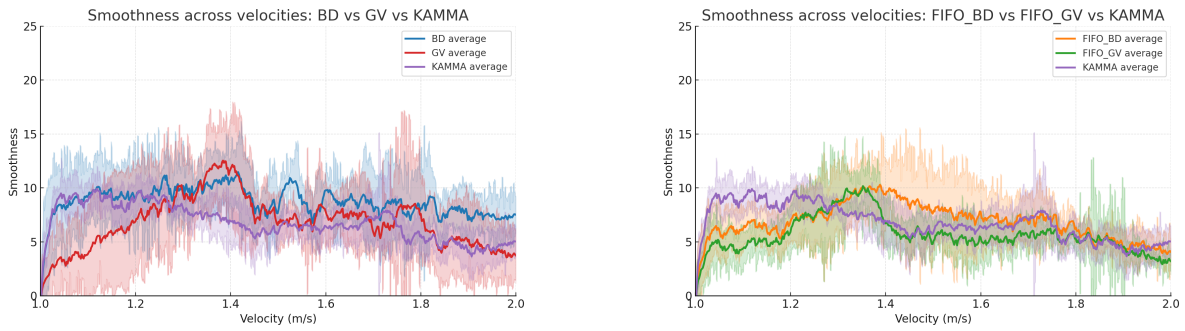


Figure 5.9: Smoothness over velocity ramp for Growing Variance (GV) and Bimodal Drift (BD) curricula (five seeds each). vs their FIFO counter parts compared to the KAMMA baseline

(from Section 5.3) tended to have higher jerk by that point. This indicates the curriculum helped the agent generalize its smooth control to the novel low-speed region better than one-shot training did.

The addition of FIFO replay further influences smoothness. Both GV+FIFO and BD+FIFO variants show very low and consistent smoothness values across the ramp, seemingly outperforming their counterparts without FIFO in terms of jerk stability. In particular, GV+FIFO achieves an extremely flat smoothness curve (indicating uniformly smooth control at all speeds), and BD+FIFO, while still not as good as GV+FIFO, is smoother than BD alone. This means that the FIFO mechanism helped the agent maintain stable torque changes, likely by concentrating learning on the current speed range and reinforcing those patterns. We expected FIFO to be counterproductive, and instead the FIFO variants delivered the lowest jerk overall. This suggests that forgetting of earlier training (which might cause relearning jitters) was reduced. Interestingly, the fact that BD+FIFO's smoothness is improved over BD means that FIFO memory retention can mitigate some negatives of a chaotic curriculum, but cannot fully erase them. Overall, from a smoothness perspective, the best performer is GV+FIFO, followed by BD+FIFO, then GV, and lastly BD. This ranking supports the idea that a well-structured curriculum (GV) is more critical than the replay strategy for smooth control, but using FIFO on top of a good curriculum can yield an extra boost in stability.

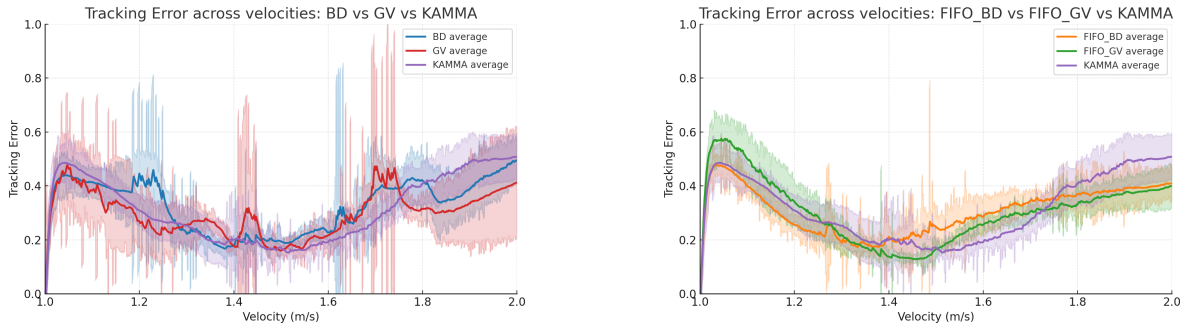


Figure 5.10: Tracking error over velocity ramp for curriculum variants: (a) GV, BD replay, (b) GV + FIFO, (d) BD + FIFO replay (five seeds each). against KAMMA baseline

Tracking Error – Figure 5.10: Turning to tracking accuracy, we observe a mix of expected outcomes and a few surprises. Starting with the GV curriculum, the GV-trained agent demonstrates generally strong tracking performance, especially at the high end of the speed range. As anticipated, GV yields noticeably better tracking at high velocities compared to a non-curriculum agent, the GV error curve rises more gently in the 1.8–2.0 m/s range, indicating improved high-speed control. This is a direct benefit of the curriculum: by the time the GV agent faces the fastest speeds, it has been through a smooth ramp-up of difficulty, so it can handle those conditions with less error. We also expected that around the Controller's working point ($\pm$1.4–1.5 m/s), the curriculum agents should have low error (since the early training and expert guidance heavily covered that range). However, an unexpected result is that the GV agent shows large error peaks even near 1.4 m/s in Figure 5.10. In fact, all four variants (GV, BD, and their FIFO versions) exhibit at least one seed with a spike of high error around the mid-speed region. This is puzzling because we would normally expect the best tracking performance at the mid-range speeds that were well represented in the training curriculum. The presence of error spikes at 1.4 m/s suggests that some forgetting or regression occurred for that nominal regime. This could be because once the curriculum moved beyond that speed, the agent focused on newer speeds and partially forgot how to excel exactly at 1.4 m/s. It could also be an artifact of the finite replay buffer (for FIFO cases) or simply variability across seeds. In any case, aside from these anomalies, GV's overall tracking error is lower and more stable than BD's, which meets our original expectation that GV would facilitate better tracking.

The BD curriculum shows slightly larger tracking performance relative to GV. The BD agents have higher average error throughout the ramp and particularly erratic behavior in at least one of the runs (reflected by wide error bands and spikes throughout the entire range). This was expected: the lack of a smooth progression makes it harder for the agent to fine-tune its velocity tracking. The oscillating

training regime seems to result in inconsistent policy behavior, as evidenced by occasional large errors. For example, a BD-trained agent might do well on either low or high speeds in isolation, but when tested on the continuous ramp (which covers the whole range), it can falter unexpectedly at some intermediate points. This confirms the intuition that task volatility in training translates to performance volatility in execution. In summary, BD variants tend to have higher tracking error and more unpredictability, whereas GV variants achieve lower error and smoother error curves, especially at the high end.

Incorporating the FIFO replay buffer had some nuanced effects on tracking. We expected FIFO to help maintain low error on the latest trained tasks while sacrificing earlier ones, but the results suggest a complicated trade-off. The GV+FIFO agent emerges as the best overall in tracking accuracy – it has the lowest errors across most of the ramp (aside from the aforementioned mid-speed blip) and particularly strong performance at high speeds. This indicates that combining GV with a recency-focused replay yields the highest tracking precision, as one might hope. The BD+FIFO agent, however, does not show a clear improvement over BD alone; in fact, BD+FIFO performed slightly worse in some portions of the range (its variance is somewhat higher than BD's at 1.3–1.6 m/s). This is an expected outcome: one assumes prioritizing recent BD tasks would amplify forgetting of the alternating contexts, leading to even poorer integration of the two task types (high vs. low commanded velocity). Essentially, because BD already causes the agent to oscillate focus, the FIFO's tendency to dump "old" data may have caused the agent to overfit to whichever context was last seen and lose proficiency in the other. The result is a slight degradation in overall BD tracking performance with FIFO, showing that memory schemes can be counterproductive if they bias too hard towards recent data in a rapidly switching task scenario.

In order to verify and identify the true mechanism of forgetting more seeds, more decay schedules, and other RB's should be tested. Our FIFO was set to hold 10,000 transitions (about 20% of total training data), this size might not have been optimal. Future tuning of the buffer size or sampling strategy could potentially reduce those mid-range error spikes.

5.4.3. Lessons on Memory, Curriculum, and Forgetting

As shown in Table 5.4.3, curriculum learning produced the best overall outcomes when combined with appropriate memory retention mechanisms. The GV curriculum led to improved final performance, achieving a lower tracking error (0.301) than the BD schedule (0.328) and yielding smoother control (smoothness 6.84 vs 8.84). GV's gradual task progression enabled the agent to converge during training ($\pm$2000 episodes) with only moderate instability (stability rated "Medium"), whereas the more abrupt, alternating BD curriculum induced convergence failures (no clear convergence within the training episodes) and pronounced instability at high speeds (stability "Low"). Incorporating a FIFO replay buffer further enhanced the GV curriculum's results. For instance, GV+FIFO attained the lowest smoothness value (5.68) and maintained stability without regressing on earlier skills, by accelerating early convergence and mitigating forgetting of recent tasks. However, even with FIFO, the BD variant remained fragile, highlighting that frequent context-switching can disrupt learning momentum and cause the agent to forget previously mastered sub-tasks. Overall, these experiments show that a well-structured curriculum (particularly GV) can expand the policy's high-speed competence beyond the base KAMMA training, but they also underscore the importance of memory-aware strategies to preserve stability and consistency when staging learning tasks.

Variant	Tracking Error ($\pm$)	Smoothness ($\pm$)	Conv. Ep.	Stability
GV	0.301 $\pm$ 0.170	6.84 $\pm$ 4.53	2000	Medium
BD	0.328 $\pm$ 0.148	8.84 $\pm$ 3.49	-	Low
FIFO GV	0.310 $\pm$ 0.136	5.68 $\pm$ 2.88	2000	Medium
FIFO BD	0.312 $\pm$ 0.113	6.91 $\pm$ 3.30	-	Low

Table 5.3: Tracking Error and Smoothness ($\pm$ std), convergence episode, and qualitative stability for Section 5.4 (GV, BD, FIFO GV, FIFO BD).

6

Discussion, Limitations, and Outlook

6.1. Key Empirical Insights

Our findings confirm several key empirical insights about integrating expert knowledge and curriculum learning into reinforcement learning (RL) for skid-steer control. First, there is a clear trade-off between control fidelity and learning difficulty when choosing the action space. Richer action representations (e.g., full 4-dimensional independent wheel torques) offer finer control and capture more of the robot's true dynamics, but they proved significantly harder to train. In our experiments, a lower-dimensional torque action space (e.g., 1D or 2D aggregated wheel torques) converged faster and more reliably, whereas the full 4D action space often required substantially more training and occasionally failed to converge under the baseline algorithm. This highlights the importance of thoughtfully selecting or structuring the action space for efficient learning (in line with observations by Eßer et al., 2024).

Second, the Knowledge-Assisted Mixed Mode Actioning (KAMMA) strategy significantly improved training stability and performance compared to the baseline Knowledge-Assisted DDPG method (Dai et al., 2022). Instead of blending expert and learner actions continuously, KAMMA uses a probabilistic switching mechanism: at each time step either the expert's action or the agent's action is executed in full, with a decaying probability of using the expert. This discrete expert injection preserved the benefits of expert guidance in early training (preventing the agent from entering irrecoverable states), while avoiding the interference and torque saturation issues that continuous blending caused. Empirically, KAMMA yielded smoother learning curves and higher final tracking accuracy than the original approach, especially in the challenging 4D action-space scenario. This confirms that strategically handing off control from expert to learner (analogous to an ϵ-greedy exploration aid) can boost the learning outcome.

Third, incorporating curriculum learning proved beneficial for both training stability and final performance. We devised and tested multiple curricula that gradually increase task difficulty, notably a monotonic Growing Variance (GV) curriculum (expanding the speed range in stages) versus a non-monotonic Bimodal Drift (BD) curriculum (alternating between easy and hard regimes), with and without a modified FIFO replay buffer to mitigate forgetting. The results showed that well-structured curricula lead to stable learning and lower tracking errors at high speeds compared to training on the full task from scratch. In particular, the progressively challenging GV curriculum enabled agents to achieve significantly better high-velocity tracking by the end of training (validating the benefit of staged difficulty shaping), whereas the oscillating BD schedule led to more erratic performance as expected. Curriculum-enhanced agents overall outperformed those trained with no curriculum, underscoring that a staged training regime can extend the effective control envelope of the learned policy.

Fourth, we found a strong synergy between expert guidance and curricula. The unified approach—using KAMMA in conjunction with a curriculum—consistently outperformed using either expert demonstrations or curricula alone. An agent that both received expert assistance and progressed through staged tasks learned faster and attained higher final accuracy than agents with only one of those aids. For example, a KAMMA-trained agent following the GV curriculum achieved better top-speed tracking than a KAMMA agent trained on a non-curated mix of tasks. This demonstrates that expert knowledge

and curricula are complementary: early expert oversight provides a strong initial bias and prevents catastrophic failures, while the curriculum ensures the agent is gradually challenged and learns to handle the full range of operating conditions, albeit with some additional stability challenges that require careful management (as later discussed in our results). Overall, our integrated training framework yielded the best performance among all variants tested.

Finally, these insights were supported by a thorough experimental evaluation. We conducted side-by-side comparisons against baselines and ablation cases (e.g. with and without KAMMA or curriculum) to isolate each component's contribution. We also tested the trained policies on unseen trajectory profiles to assess generalization. The KAMMA-based policies notably surpassed pure imitation-learning agents in challenging scenarios, and curriculum-trained agents maintained higher tracking precision at extreme speeds than those without curricula. Taken together, our results validate the central hypothesis that combining staged tasks with expert-assisted RL produces more robust and accurate control policies than either approach on its own.

6.2. Methodological Surprises

Beyond the planned outcomes, the study revealed a few unexpected findings and nuances in our methodology. One noteworthy surprise was an anomaly in the curriculum agent's performance at mid-range speeds. We had anticipated the lowest tracking errors in the speed range that was heavily covered during early training. However, some curriculum-trained agents exhibited unexplained error spikes around the expert controller's nominal speed (approximately 1.4 m/s), even as their high-speed tracking improved. This counter-intuitive dip in performance suggests a form of forgetting or regression: once the curriculum progressed beyond that medium-speed regime, the agent seemingly forgot some of its proficiency at the earlier stage. Possible causes include the agent focusing on new, faster regimes at the expense of mid-speed mastery, or limitations of the replay buffer (even in variants that used a FIFO buffer to retain recent experience). Although overall performance was still better with the curriculum than without, this finding highlights a subtle challenge in staged training—agents may temporarily regress on earlier subtasks when those are no longer emphasized. Recognizing this helped us identify the need for mechanisms to maintain performance on intermediate goals (e.g. revisiting earlier tasks or using a more adaptive curriculum schedule).

Another unexpected issue was the sensitivity introduced by reward shaping and schedule tuning. In our framework, the expert's involvement and the curriculum progression are governed by hand-tuned schedules (for fading out expert assistance and advancing task difficulty). We observed that if the shaping (expert-assisted) reward component is not carefully balanced, it can mask the agent's true performance early in training. For instance, an agent might continue to receive high total rewards due to the residual expert-driven reward, giving a false impression of improvement even if its own policy has plateaued. Only once the expert influence diminishes would the underlying performance drop become evident – at which point the sudden loss of the shaping reward can cause instability in learning. This phenomenon, along with the general variability introduced by different schedule parameters, was more pronounced than expected. It underlines the importance of careful schedule design: even small adjustments to the expert decay rate or curriculum step criteria significantly affected convergence speed and stability. In retrospect, this surprise reinforced the value of monitoring both shaped and true reward signals during training, and motivated exploring more adaptive scheduling techniques to eliminate such brittle behavior.

6.3. Limitations and Mitigation Strategies

While the results are encouraging, several limitations must be acknowledged, along with potential strategies to mitigate them:

- **Sim–to–Real Gap:** All training and evaluation in this study were conducted in high-fidelity simulation. Despite our efforts to use a realistic physics model, the learned policy has never been exposed to real-world dynamics (e.g., exact friction variations, compliance, or unmodeled mechanical effects). This raises uncertainty about how well the policy would transfer to a physical skid-steer robot, subtle factors like tire deformation or drivetrain backlash could degrade performance. Furthermore, we ran the policy at 10 Hz, which was essential for obtaining consistent

learning behavior. This relatively low control frequency provided stability by allowing the agent to observe the longer-term effects of its actions, especially in relation to traction and slip. However, this setup also imposes a limit on the environments we can handle. If conditions become more dynamic, such as lower friction coefficients, higher velocities, or abrupt terrain changes, a faster control loop might be necessary to ensure stability and responsiveness. Even for expert controllers. This suggests that while 10 Hz sufficed for our straight-line, flat-ground experiments, different tasks or real-world implementations may demand adaptive control frequencies. *Mitigation:* Bridging this sim-to-real gap will require careful field trials and possibly additional system identification. Techniques such as domain randomization (varying simulator parameters within realistic bounds) and adding sensors or feedback for traction estimation can help the policy learn to handle real-world variability. Ultimately, validating and fine-tuning the controller on physical hardware is essential to ensure that the learned policy remains robust under real operating conditions.

- **Manual Schedule Sensitivity:** Our approach relies on manually-designed curricula and expert intervention schedules. As noted above, the performance can be sensitive to these scheduling parameters – a curriculum broken into different stage lengths, or a slightly faster/slower expert handoff, can lead to noticeably different outcomes in convergence speed or stability. The need for extensive tuning not only increases development effort but also poses a reproducibility challenge: what works optimally in our setup might falter if the timing or thresholds are altered. *Mitigation:* A promising solution is to incorporate adaptive or automated schedule tuning. Future implementations could use performance-based triggers or meta-learning to adjust the difficulty progression and expert assistance in real time. For example, the agent's recent error rate or learning progress could dictate when to escalate task difficulty or reduce expert input, instead of adhering to a fixed schedule. By making the curriculum and expert handover self-tuning, we can improve the method's robustness and reduce the burden of manual parameter tuning. This would likely alleviate the schedule-sensitivity issue and make it easier for others to replicate our results without exhaustive hand-tuning.
- **Domain Specificity:** We intentionally constrained our experiments to a narrow domain – straight-line trajectory tracking on flat terrain using a single skid-steer robot model. It remains uncertain how well the learned policy and insights generalize to other scenarios. Real deployments may involve uneven terrains (slopes, rough or deformable ground), different maneuvers (sharp turns, obstacle avoidance), and varying vehicle configurations or weights. Our current results do not guarantee success in those conditions. *Mitigation:* To broaden the applicability, the approach should be tested and extended in more diverse settings. Incorporating terrain-aware features (e.g. sensing inclines or surface friction) into the state could allow the policy to anticipate changing conditions. Combining this with domain randomization during training (varying ground traction, robot parameters, and environmental conditions) would expose the agent to a wider range of situations, likely improving its generalization. Follow-up studies should evaluate the controller on complex terrains and with different robot hardware to identify any failure modes and adjust the method accordingly. Expanding the domain in this way will help map out the boundaries of our approach's validity and ensure it is not limited to just the nominal scenarios we studied.
- **Sample Efficiency and Scalability:** The computational cost of training our agents was high, reflecting a limitation in sample efficiency. Each agent required many hours of simulated experience on a GPU to converge, and by extrapolation this would translate to several days of real-world training per agent (which is impractical and would cause considerable wear on a real robot). Moreover, our hyperparameters were largely adopted from prior work and not exhaustively optimized for each variant; it is possible that with further tuning, performance could improve or training time reduce, but finding those optimal settings itself is resource-intensive. This scalability issue means that deploying the method in the field or on larger tasks would be challenging. *Mitigation:* Future research should prioritize improving learning efficiency. Potential avenues include integrating model-based elements or simulators to guide exploration, using more advanced replay buffer strategies (to reuse past experiences more effectively and prevent forgetting), and parallel or meta-learning approaches to tune hyperparameters automatically. Reducing the training time by even an order of magnitude would greatly enhance the practicality of the approach, both for simulation studies and real-world trials. In addition, techniques like hierarchical learning

(discussed below) could break the task into simpler subtasks, making learning more tractable. By addressing the sample inefficiency and scaling limitations, we move closer to a framework that is not only effective but also feasible to implement outside of controlled simulation environments.

6.4. Broader Research Opportunities

Building on this work, there are several broader research opportunities to explore. One important direction is automating the curriculum and expert scheduling. Developing agents that can learn to regulate their own training schedule would make the approach more adaptive and reduce manual intervention. For example, an meta-learning algorithm could adjust the progression of task difficulty or the reliance on expert actions based on the agent's performance in real time. Such adaptive curricula and self-tuning expert guidance mechanisms would improve reproducibility and might yield even faster convergence by continuously challenging the agent at an appropriate level.

A second avenue is to extend the framework to more complex tasks and environments. Future research can introduce richer and more variable scenarios to push the limits of the learned controller. This includes testing on uneven or changing terrains (such as slopes, rough ground, or mixed-friction surfaces) and handling more dynamic maneuvers (like curving paths or obstacle avoidance). To succeed in these settings, the policy may need additional inputs or modules – for instance, incorporating exteroceptive sensors (lidar, vision or inertial measurements) to perceive upcoming terrain features, or conditioning the policy on estimated friction coefficients. Coupling these enhancements with extensive domain randomization during training (randomly varying physical parameters, terrains, and disturbances) will help the agent acquire a robustness to environmental changes. By broadening the training domain in this way, we can assess how general the current approach is and adapt it for real-world complexity, thus narrowing the gap between our simulation setup and practical field conditions.

Another promising research direction is Hierarchical Reinforcement Learning (HRL) for skid-steer control. Our current approach treats the task at a single level, but as we consider more complicated skills or longer-horizon goals, a flat policy may struggle with the "curse of dimensionality." HRL could introduce multiple layers of controllers: a high-level policy could make strategic decisions or select subtasks (for example, choosing between a "precision low-speed" mode vs. a "high-speed maneuvering" mode, or setting intermediate waypoints), while low-level policies execute the fine-grained torque control to achieve those sub-goals. Such a decomposition can simplify learning by allowing each policy to focus on a narrower subproblem. The higher-level policy would handle abstract decision-making over longer timescales, and the lower-level ones would specialize in short-term dynamics. This hierarchy not only could improve learning efficiency (by reusing low-level skills across different contexts) but also manage complexity better, as evidenced by the success of staged training in our work. Incorporating an HRL framework builds on the intuition behind our curricula – breaking the challenge into stages – and could enable the system to tackle more elaborate control tasks than a single monolithic policy could feasibly learn.

Finally, a critical step forward is to validate and refine the approach on real robotic hardware. Deploying the full KAMMA + curriculum framework on an actual skid-steer vehicle will test the true robustness of the learned controller. Transitioning from simulation to reality will require addressing practical issues such as sensor noise, state estimation errors, actuation latency, and the safety of exploration (to avoid harming the robot). Additional safeguards and adaptations should be investigated to facilitate this transfer. Possible strategies include on-line system identification (continually estimating parameters like friction or slippage in real time and feeding that into the controller), using more uncertainty-aware policies (e.g. Bayesian approaches or ensemble methods that can recognize when the robot is in unfamiliar states), and leveraging a mixture of simulated and real data for fine-tuning (for example, seeding the replay buffer with some real-world experiences to ground the learning). Techniques for direct sim-to-real transfer, such as domain adaptation or adding a transfer loss in training, could also be beneficial to minimize reality gap issues. Successfully running our learned policy on a physical robot – reliably tracking aggressive trajectories under real-world conditions – would be the ultimate proof of concept for this research. It would also provide invaluable feedback on failure modes that only manifest on hardware, guiding further improvements. In summary, by pursuing these opportunities (greater adaptivity in training, expansion to complex tasks, hierarchical control structures, and real-world deployment), future work can build upon the foundations laid in this thesis and move closer to

agile, reliable learning-based control strategies for skid-steer robots operating outside of simulation.

6.5. Conclusion

In conclusion, this research has demonstrated the potential of combining curriculum learning with expert-assisted reinforcement learning to improve the data-efficiency and performance of a torque-controlled skid-steer robot. We presented a unified approach that integrates action-space design, staged task curricula, and intermittent expert guidance, and we showed that each of these components contributes to more robust learning and more precise trajectory tracking than standard RL baselines. At the same time, we have identified key limitations – such as schedule sensitivity, domain specificity, and sim-to-real transfer challenges – that temper these successes. The contributions of this work provide a solid foundation and proof-of-concept for learning-driven skid-steer control, but realizing their full potential will require continued refinement and validation under real-world conditions. By addressing the outlined limitations and exploring the proposed research directions, future investigators can further generalize and strengthen this approach. Ultimately, we believe that such efforts will bring us closer to deploying adaptive, high-performance RL-based controllers on skid-steer robots, enabling them to reliably execute agile maneuvers in the field.

Appendix

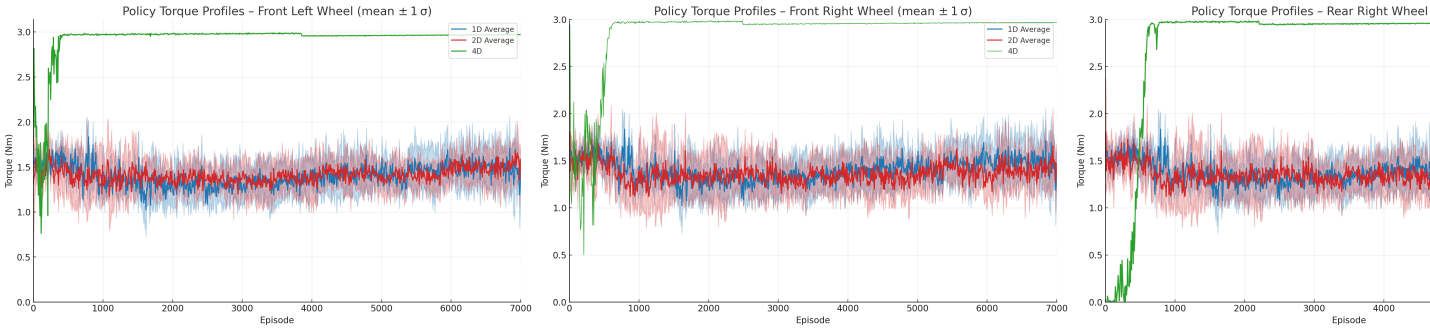


Figure 6.1: Front Left, Front Right, Rear Left, and Rear Right torque profiles during training for 1D (blue), 2D (red), and 4D (green) KA-DDPG variants (five seeds each).

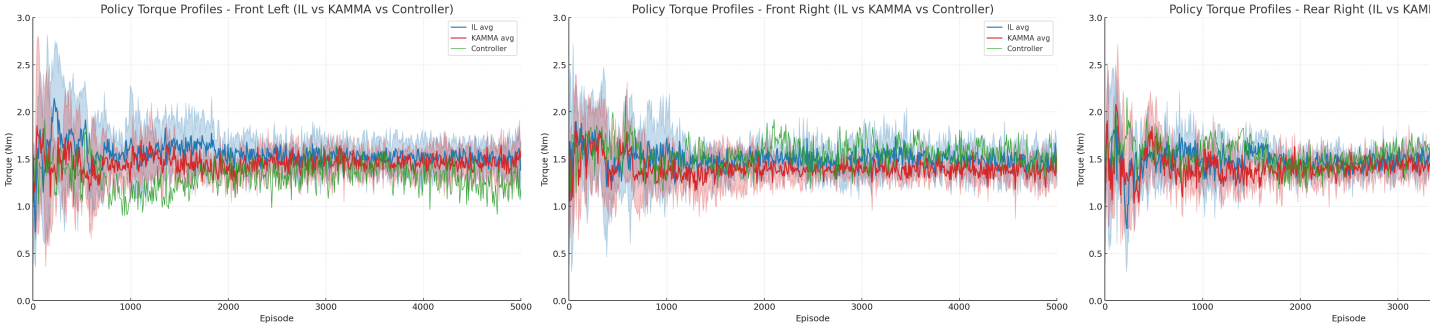


Figure 6.2: Front Left, Front Right, Rear Left, and Rear Right torque profiles during training for IL (blue), KAMMA (red) variants (five seeds each), and the controller (green).

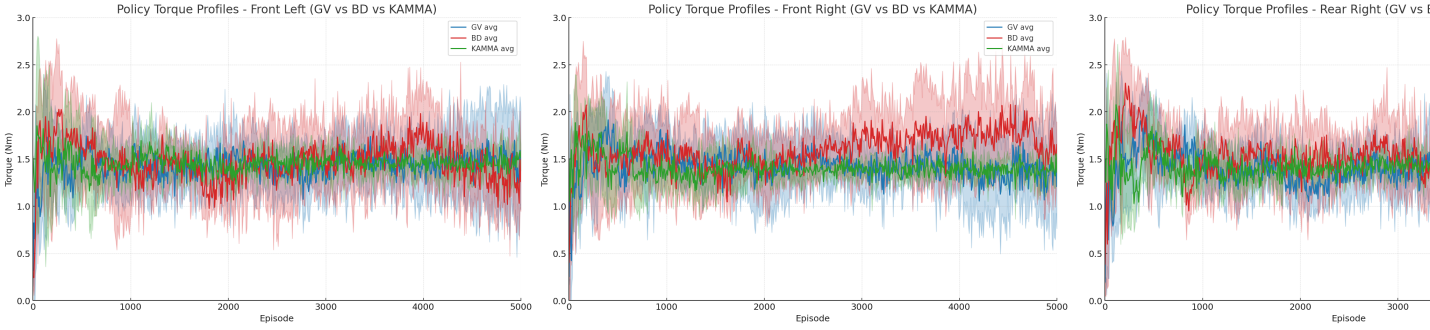


Figure 6.3: Front Left, Front Right, Rear Left, and Rear Right torque profiles during training for GV (blue), BD (red), and KAMMA (green) variants (five seeds each).

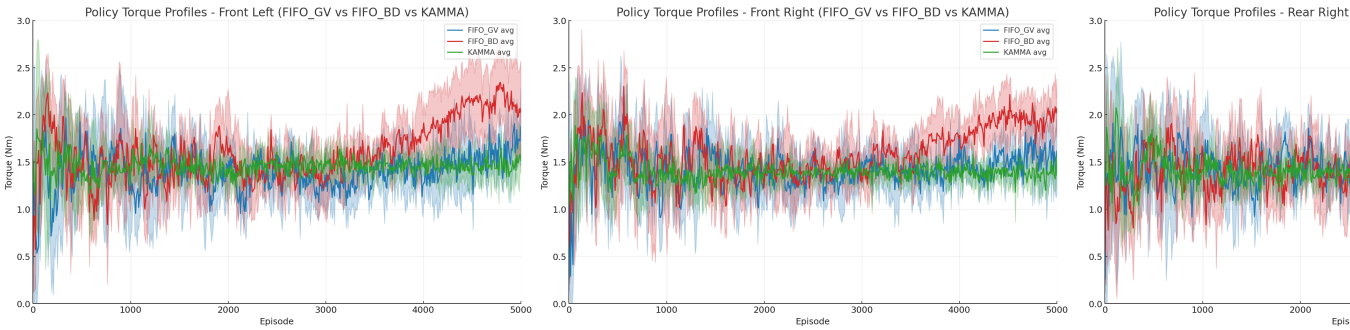


Figure 6.4: Front Left, Front Right, Rear Left, and Rear Right torque profiles during training for FIFO GV (blue), FIFO BD (red), and KAMMA (green) variants (five seeds each).

Bibliography

Chamorro, S., Klemm, V., Valls, M. d. l. I., Pal, C., & Siegwart, R. (2024). Reinforcement Learning for Blind Stair Climbing with Legged and Wheeled-Legged Robots. http://arxiv.org/abs/2402.06143

Chivkula, P., Rodwell, C., & Tallapragada, P. (2022). Curriculum-based reinforcement learning for path tracking in an underactuated nonholonomic system. *IFAC-PapersOnLine*, *55*(37), 339–344. https://doi.org/10.1016/j.ifacol.2022.11.207

Cimurs, R., & Merchán-Cruz, E. A. (2022). Leveraging Expert Demonstration Features for Deep Reinforcement Learning in Floor Cleaning Robot Navigation. *Sensors (Basel, Switzerland)*, *22*(20). https://doi.org/10.3390/s22207750

Coletti, C. T., Williams, K. A., Lehman, H. C., Kakish, Z. M., Whitten, D., & Parish, J. J. (2023). Effectiveness of Warm-Start PPO for Guidance with Highly Constrained Nonlinear Fixed-Wing Dynamics. *2023 American Control Conference (ACC)*. https://doi.org/10.23919/ACC55779.2023.10156267

Dai, H., Chen, P., & Yang, H. (2022). Driving Torque Distribution Strategy of Skid-Steering Vehicles with Knowledge-Assisted Reinforcement Learning. *Applied Sciences (Switzerland)*, *12*(10). https://doi.org/10.3390/app12105171

Datar, A., Pan, C., Nazeri, M., & Xiao, X. (2024). Toward Wheeled Mobility on Vertically Challenging Terrain: Platforms, Datasets, and Algorithms. *Proceedings - IEEE International Conference on Robotics and Automation*, 16322–16329. https://doi.org/10.1109/ICRA57147.2024.10610079

Eßer, J., Margolis, G. B., Urbann, O., Kerner, S., & Agrawal, P. (2024). Action Space Design in Reinforcement Learning for Robot Motor Skills. *OpenReview*.

Han, T., Shah, P., Rajagopal, S., Bao, Y., Jung, S., Talia, S., Guo, G., Xu, B., Mehta, B., Romig, E., Scalise, R., & Boots, B. (2025). Demonstrating Wheeled Lab: Modern Sim2Real for Low-cost, Open-source Wheeled Robotics. http://arxiv.org/abs/2502.07380

Hoeller, D., Rudin, N., Sako, D., & Hutter, M. (2023). ANYmal Parkour: Learning Agile Navigation for Quadrupedal Robots. *Science Robotics*. http://arxiv.org/abs/2306.14874

Hu, Y., Xie, Q., Jain, V., Francis, J., Patrikar, J., Keetha, N., Kim, S., Xie, Y., Zhang, T., Fang, H.-S., Zhao, S., Omidshafiei, S., Kim, D.-K., Agha-mohammadi, A.-a., Sycara, K., Johnson-Roberson, M., Batra, D., Wang, X., Scherer, S., … Bisk, Y. (2023). Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis. http://arxiv.org/abs/2312.08782

Liessner, R., Schroer, C., Dietermann, A., & Bäker, B. (2018). Deep reinforcement learning for advanced energy management of hybrid electric vehicles. *ICAART 2018 - Proceedings of the 10th International Conference on Agents and Artificial Intelligence*, *2*, 61–72. https://doi.org/10.5220/0006573000610072

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. http://arxiv.org/abs/1509.02971

Margolis, G. B., Yang, G., Paigwar, K., Chen, T., & Agrawal, P. (2022). Rapid Locomotion via Reinforcement Learning. http://arxiv.org/abs/2205.02824

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., & Stone, P. (2020). Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. http://arxiv.org/abs/2003.04960

Ostafew, C. J., Schoellig, A. P., & Barfoot, T. D. (2014). Learning-Based Nonlinear Model Predictive Control to Improve Vision-Based Mobile Robot Path-Tracking in Challenging Outdoor Environments. *2014 IEEE International Conference on Robotics & Automation (ICRA)*, 6822. https://doi.org/10.1109/ICRA.2014.6907444

Rudin, N., Hoeller, D., Reist, P., & Hutter, M. (2021). Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning. *arXiv*. http://arxiv.org/abs/2109.11978

Salimpour, S., Peña-Queralta, J., Paez-Granados, D., Heikkonen, J., & Westerlund, T. (2025). Sim-to-Real Transfer for Mobile Robots with Reinforcement Learning: from NVIDIA Isaac Sim to Gazebo and Real ROS 2 Robots. http://arxiv.org/abs/2501.02902

Scomparin, L., Becker, J., Blomley, E., Bründermann, E., Caselle, M., Dritschler, T., Kopmann, A., Müller, A.-S., Santamaria Garcia, A., Steinmann, J. L., & Xu, C. (2024). *Real-time Reinforcement Learning on AI Engines with Online Training for Autonomous Accelerators* (tech. rep.). www.kit.edu

Silver, D., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2015). *Deterministic Policy Gradient Algorithms* (tech. rep.).

Srikonda, S., Norris, W. R., Nottage, D., & Soylemezoglu, A. (2022). Deep Reinforcement Learning for Autonomous Dynamic Skid Steer Vehicle Trajectory Tracking. *Robotics*, *11*(5). https://doi.org/10.3390/robotics11050095

Sutton, R. S. (2019). The bitter lesson. http://www.incompleteideas.net/IncIdeas/BitterLesson.html

Sutton, R. S., & Barto, A. G. (2018). Policy Gradient Methods. In *Reinforcement learning: An introduction* (2nd ed., pp. 321–339). MIT Press.

Tsampazis, K., Kirtas, M., Tosidis, P., Passalis, N., & Tefas, A. (2023). Deep Reinforcement Learning With Action Masking for Differential-Drive Robot Navigation Using Low-Cost Sensors. *IEEE International Workshop on Machine Learning for Signal Processing, MLSP*, *2023-September*. https://doi.org/10.1109/MLSP55844.2023.10285997

Wiberg, V., Wallin, E., Servin, M., & Nordfjell, T. (2021). Control of rough terrain vehicles using deep reinforcement learning. http://arxiv.org/abs/2107.01867

Xie, Z., Lin, Z., Li, J., Li, S., & Ye, D. (2022). Pretraining in Deep Reinforcement Learning: A Survey. http://arxiv.org/abs/2211.03959

Xu, T., Pan, C., & Xiao, X. (2024). Reinforcement Learning for Wheeled Mobility on Vertically Challenging Terrain. http://arxiv.org/abs/2409.02383