RobotxR1: Enabling Embodied Robotic Intelligence on Large Language Models through Closed-Loop Reinforcement Learning

Liam Boyle[†,*], **Nicolas Baumann**[†,‡,*], **Pavithithiren Sivasothilingam**[†], **Michele Magno**[†], **Luca Benini**[‡]
[†]Center for Project-Based Learning, [‡]Integrated Systems Laboratory
ETH Zurich

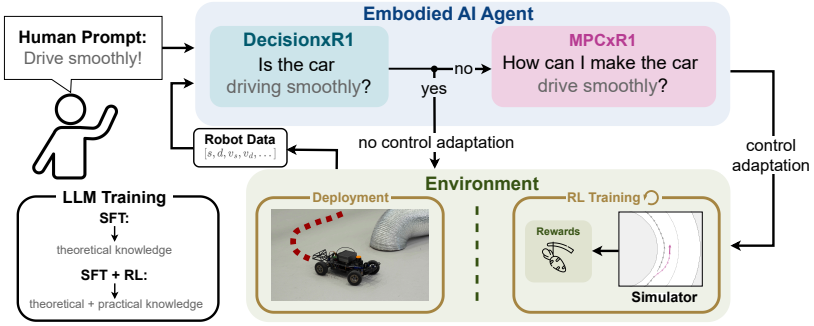

Figure 1: Overview of the proposed Embodied AI agent for autonomous driving. The agent consists of a *DecisionxR1* module and an *MPCxR1* module that work in tandem to achieve a user-specific driving behavior. The LLMs at the core of each module are trained with SFT and RL. During RL training, the system interacts with its simulation environment, where it is rewarded for behavioral adherence. The trained agent is deployed on a scaled autonomous car for real-world experiments.

Abstract: Future robotic systems operating in real-world environments will require on-board embodied intelligence without continuous cloud connection, balancing capabilities with constraints on computational power and memory. This work presents an extension of the *R1-zero* approach, which enables the usage of low parameter-count Large Language Models (LLMs) in the robotic domain. The *R1-Zero* approach was originally developed to enable mathematical reasoning in LLMs using static datasets. We extend it to the robotics domain through integration in a closed-loop Reinforcement Learning (RL) framework. This extension enhances reasoning in Embodied Artificial Intelligence (Embodied AI) settings without relying solely on distillation of large models through Supervised Fine-Tuning (SFT). We show that small-scale LLMs can achieve effective reasoning performance by learning through closed-loop interaction with their environment, which enables tasks that previously required significantly larger models. In an autonomous driving setting, a performance gain of 20.2%-points over the SFT-based baseline is observed with a *Qwen2.5-1.5B* model. Using the proposed training procedure, *Qwen2.5-3B* achieves a 63.3% control adaptability score, surpassing the 58.5% obtained by the much larger, cloud-bound *GPT-4o*. These results highlight that practical, on-board deployment of small LLMs is not only feasible but can outperform larger models if trained through environmental feedback, underscoring the importance of an interactive learning framework for robotic Embodied AI — one grounded in practical experience rather than static supervision.

Keywords: Large Language Models, Reinforcement Learning, Embodied AI, Constrained Hardware

[*]Denotes qual contribution. Code at: github.com/ForzaETH/LLMxRobot

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

1 Introduction

The robotics domain has seen significant progress through data-driven Machine Learning (ML) approaches, where the prevailing strategy has been to construct large-scale datasets [1, 2, 3, 4, 5] to train increasingly large and complex Neural Networks (NNs) [6, 7, 8, 9, 10, 11] in a supervised manner. The underlying assumption is that larger datasets provide greater task and environmental diversity, reducing the likelihood that robots encounter unfamiliar scenarios at deployment [7, 11]. However, real-world environments are inherently unpredictable and contain numerous edge cases that are intractable to fully capture in any dataset [12, 13]. Humans rely on prior knowledge and contextual reasoning to recognize anomalous situations and adapt their behavior accordingly [14, 15, 16]. To date, Large Language Models (LLMs) represent the closest approximation to artificial knowledge-based systems, making their integration into robotic systems a promising direction for Embodied Artificial Intelligence (Embodied AI) [14, 16, 17].

The recent release of Guo et al. [18], introducing the *DeepSeek R1-Zero* method, marks a significant advancement in enabling reasoning within LLMs [18, 19]. The approach integrates Reinforcement Learning from Verifiable Rewards (RLVR) with LLMs applied for mathematical datasets such as `AIME 2024`, `MATH-500`, or `GSM8K` [20], and has sparked discussion regarding the extent to which such models are capable of solving problems considered to be at PhD level in mathematics [21, 22]. However, this required training a massive LLM within a Reinforcement Learning (RL) loop — an approach that incurs extremely high computational costs [18, 23, 24]. In this work, we focus on Autonomous Driving Systems (ADS) as a concrete and high-stakes example of embodied robotic intelligence, where deploying such large models poses a major limitation, as it prevents edge deployment without reliance on cloud infrastructure. Specifically for ADS and other robotic domains, ubiquitous cloud connectivity is infeasible and introduces serious security vulnerabilities [25, 26, 27].

One potential solution is to distill the reasoning capabilities of large LLMs into smaller, more efficient models through Supervised Fine-Tuning (SFT), an approach also demonstrated effectively in the *R1-Zero* framework [18]. Distilled models may inherit some reasoning ability, but unlike human reasoning, which is tightly coupled with sensory feedback, situational awareness, and physical context, these models operate in abstraction, lacking the closed-loop perception-action embodiment that underpin robust robotic intelligence [28]. Moreover, RLVR was first introduced to operate on static math or coding datasets [18, 29], whereas RL in robotics typically involves interaction with dynamic environments [28, 30, 31]. Success in solving abstract, mathematical problems does not necessarily translate to the embodied, context-dependent reasoning needed for robotic operation. Hence, we argue that the type of reasoning required for robotics is fundamentally different from that assessed in PhD-level mathematics benchmarks. Similarly, one must ask whether ADS should rely on models trained to mimic mathematical problem-solving, or rather on systems that acquire reasoning capabilities through direct interaction with their environment.

Our contributions are fourfold: **(i)** *RobotxR1* is introduced as an extension of the *R1-Zero* framework, showing that direct interaction of LLMs with their environment via closed-loop RL is possible in an ADS setting, moving beyond static dataset RLVR training. **(ii)** Edge-deployable LLMs are shown to effectively learn through interaction with a scaled autonomous vehicle, with a 3B model achieving a 63.3% control adaptability score — surpassing the 58.5% obtained by the much larger cloud-based *GPT-4o*. **(iii)** Interaction-based training is shown to substantially benefit small models, with a 1.5B model achieving a 20.2%-point improvement over its SFT-only baseline, underscoring the value of embodied learning through interaction, in robotic Embodied AI. **(iv)** The proposed method has low computational demands, with training feasible on a single consumer-grade Graphics Processing Unit (GPU) (e.g., RTX 3090), and with the *Qwen2.5* 1.5B and 3B models being capable of deployment on an embedded Jetson Orin AGX serving as the robot's OnBoard Computer (OBC).

2 Related Work

Robot Control and LLMs: Recent research has explored the integration of LLMs with robotic control systems [14, 17, 32, 33, 34, 35]. These works consistently find that LLMs are ill-suited for direct low-level control, and instead should influence behavior indirectly, typically by shaping a reward or cost signal for a dedicated low-level controller, such as in a Model Predictive Controller (MPC) framework [32, 33]. Ismail et al. [34] propose using an LLM to generate objective functions and constraints for manipulation tasks conditioned on human prompts. Their architecture combines the flexibility of LLMs with established and classic MPC controllers. Similarly, Baumann et al. [32] introduces an open-source framework for LLM-based robot interaction through a low-level MPC, and demonstrates it on a 1:10 scaled autonomous racing car. Our work targets the same tightly constrained deployment scenario. However, existing work primarily relies on SFT or closed-source, cloud-based models like *GPT-4*, which cannot be finetuned locally and are thus limited to prompt engineering. As such, these methods lack trainable Embodied AI reasoning capability.

Reasoning with LLMs: The *R1-zero* framework introduced by Guo et al. [18] demonstrates how an LLM can be integrated into a RLVR loop for structured reasoning tasks on static datasets, such as multiple-choice math questions [29]. The reward signal combines formatting and correctness objectives, where formatting ensures the answer can be reliably parsed. This approach has shown State of the Art (SotA) reasoning performance in mathematics-based tasks. Follow-up work includes Dao and Vu [36], which applies RLVR, and more specifically Group Relative Policy Optimization (GRPO), in an *R1-zero* fashion to enable spatial reasoning in maze-solving tasks, and Azzolini et al. [37], which extends the framework to a Vision Language Model (VLM) for reasoning in embodied agents within autonomous driving scenarios. However, these methods are limited to static datasets and lack interaction with dynamic environments, missing out on streams of experiences that arise from closed-loop environmental feedback [28].

By contrast, RL in robotics typically involves continuous interaction with an environment, forming a closed feedback loop. In this work, we close this gap by adapting the *R1-zero* process to a dynamic ADS simulation environment, where the LLM interfaces with a low-level MPC controller and must reason over human driving commands to guide the robot's behavior. To the best of our knowledge, this is the first LLM-driven training loop in continuous control with direct simulation feedback, enabling real-time, language-grounded decision-making combined with classical control.

3 Methodology

This work leverages the publicly available framework introduced in [32] to enable a comparison with purely SFT-trained LLMs, while significantly extending the *DecisionxLLM* and *MPCxLLM* architectures of [32] to support robotic reasoning in dynamic environments. The existing Retrieval Augmented Generation (RAG) structure, including the use of five retrieved memories as in the baseline [32], and the proposed Low Rank Adaptation (LoRA)-based SFT method are leveraged. In this work, the *Qwen* family of models [38], ranging from 1.5 to 7B parameters, is adopted as the primary LLM architecture, motivated by the performance reported in [32]. The *R1-zero* framework is integrated into this setup, resulting in *RobotxR1*, which encompasses the *DecisionxR1* and *MPCxR1* modules, to enable full RL-based embodiment of the underlying LLMs within a holistic pipeline that follows sequential decision making and control adaptation, as illustrated in Figure 1.

3.1 Robotic Autonomy Stack and MPC

The utilized robotic autonomy stack follows the 1:10 scaled racecar [39], where the racing line corresponds to a minimum curvature trajectory computed for a closed race track. This racing line is then tracked utilizing a kinematic MPC defined in a curvilinear coordinate system as in [32]. The robot state is defined by $x = [s \quad n \quad \Delta\phi \quad \delta \quad v]^T$, where s and n represent the longitudinal and

lateral distances to the racing line; $\Delta\phi$ denotes the heading error relative to the racing line; δ is the steering angle; and v is the longitudinal velocity. The MPC control input is $u = [\Delta\delta, a]^T$.

$$\min_{u} \; J(x,u) = \sum_{i=0}^{N-1} q_n n_i^2 + q_v (v_i - v_{\text{ref}})^2 + q_\alpha \Delta\phi_i^2 + \|\Delta u_i\|_{q_R} \quad s.t. \quad x_i \in \mathcal{X}, u_i \in \mathcal{U} \tag{1}$$

Here, N denotes the prediction horizon. The weights q_n, q_α, and q_v correspond to state terms, while q_R is a diagonal matrix penalizing control inputs. State and input constraints are defined by $\mathcal{X}$ and $\mathcal{U}$. Velocity v and steering angle δ are bounded in magnitude, and lateral error n is constrained within track boundaries, modulated by an inflation factor ϵ for safety. Steering rate and acceleration are also constrained. The default MPC parameters have been empirically obtained and optimized to track the racing line effectively.

The cost function weights are exposed as online-tunable parameters, enabling real-time adaptation of driving behavior. Additionally, constraints such as for example the boundary inflation factor ϵ and the velocity bounds can be modified by the LLM, allowing the robot's driving policy to be dynamically adjusted in response to natural language instructions (more information in Appendix E).

3.2 DecisionxR1 — Enhanced Reasoning

The objective of the *DecisionxR1* module, depicted in Figure 2, is to teach an LLM to reason about the current driving behavior of an autonomous racing car. More specifically, the LLM should decide whether the car is adhering to the driving behavior prompted by the human using a given history of robot states. Following the recent successes of other works [36, 37, 40, 41] on training LLMs for embodied reasoning with RL, we adopt the two-stage *R1-zero* training procedure introduced in [18] to finetune the *DecisionxR1* module with RLVR. In the first stage, the model undergoes SFT following the methodology of [32], which distills embodiment-specific knowledge into pre-trained *Qwen* 1.5B and 3B models. In the second stage, the model is further optimized using RLVR, more specifically GRPO, guided by a static decision-making dataset and reward functions designed to reinforce both decision accuracy and output structure.

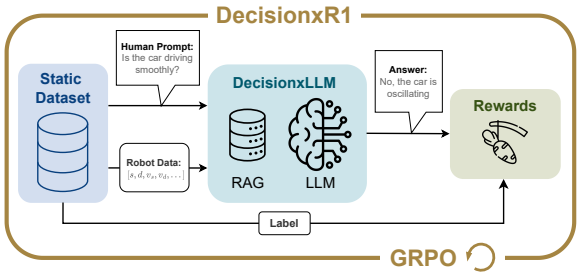

Figure 2: The DecisionxR1 module consists of a RAG enhanced LLM that uses robot state information to determine if the car is adhering to the desired behavior prompted by the human. Each answer of the LLM is assigned a format and correctness reward that are used to train the model with GRPO.

Decision-Making Dataset: Similar to [18, 36], we build a static dataset in which each instance is formulated as a binary classification task. The dataset was obtained by driving the robot in simulation in eight different driving styles (e.g., centerline tracking, reversing, raceline tracking, etc.), and recording robot state information x, as in Equation (1). The binary behavior adherence label is computed programmatically. Programmatic labeling is done by defining a set of rules in $\mathcal{B}$, that evaluate how closely the robot's actions match a target driving style. For example, for reversing, if the longitudinal velocity v is negative, the label is set to 1 (adherence), otherwise 0 (non-adherence).

Reward Modeling for Decision Making: We employ two kinds of rewards, which we roughly define as accuracy and formatting rewards, such that the total reward can be described as

$$R_{\text{DecisionxR1}} = R_{\text{accuracy}} + R_{\text{fmt}}, \tag{2}$$

where R_{accuracy} is the reward the model gets for correctly determining if the robot is adhering to the prompted behavior and R_{fmt} is a formatting reward that should encourage the model to structure its answer into reasoning, explanation, and answer sections (e.g. using `<reasoning>` and

</reasoning> tags to mark the beginning and end of the reasoning section). While the correctness reward incentivizes accurate decision-making, the formatting reward serves a dual purpose: it encourages the model to articulate its reasoning and ensures that the output remains structured and easily parsable.

3.3 MPCxR1 — Enhanced Control Adaptability

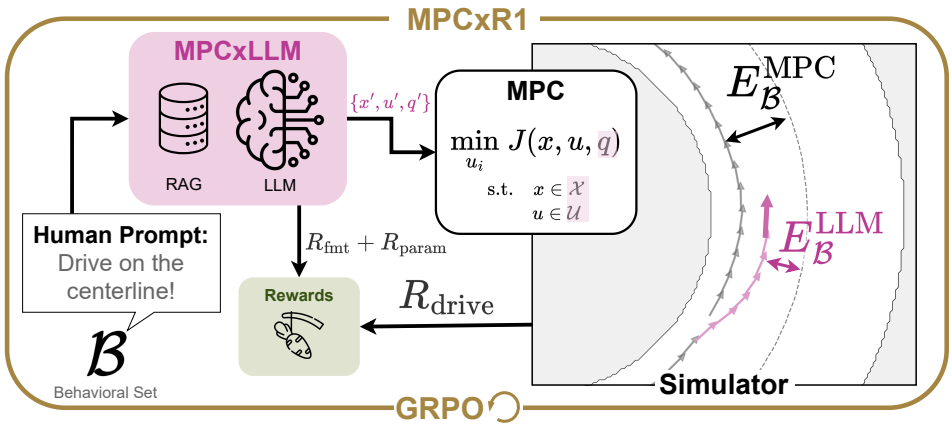

Figure 3: Schematic overview of the proposed *MPCxR1* training procedure. The LLM leverages RAG context, following [32], to generate MPC parameters tailored to the desired driving behavior $\mathcal{B}$. The LLM and MPC operate in a closed-loop with a simulator, which computes both the baseline Root Mean Square Error (RMSE) under default MPC parameters ($E_{\mathcal{B}}^{\text{MPC}}$) and the RMSE resulting from the LLM-generated parameters ($E_{\mathcal{B}}^{\text{LLM}}$), both relative to the behavioral objective, later introduced in Equation (1). These are used to compute the behavioral reward R_{drive}, while the LLM's textual output is evaluated for formatting (R_{fmt}) and parameter validity (R_{param}). This framework extends *R1-zero* from static dataset training to a fully embodied RL setup via GRPO.

In contrast to the *DecisionxR1* module introduced in Section 3.2, control adaptability in robotic systems cannot rely solely on static datasets. As in many robotic applications, effective behavior emerges through interaction with the environment rather than from parameters learned via SFT alone. As an analogy, this distinction reflects the difference between learning to drive by reading a manual (SFT) and learning through actual driving lessons (RLVR). In this setting, the *MPCxR1* module places the LLM in a closed-loop with the MPC, enabling it to influence control behavior based on interaction, thereby facilitating embodied learning and control adaptation through experience.

Closed-Loop RL Environment: As illustrated in Figure 3, the standard *R1-zero* training process is modified to incorporate closed-loop feedback from a driving simulation. The training spans a range of driving behaviors $\mathcal{B}$ (see Appendix C for the full set), with each prompt specifying a distinct behavioral objective. One such case is instructing the *MPCxR1* module to *"Drive at 1.83 m/s as closely as possible"*, as shown in Bubble 1. In response, the LLM generates MPC parameters intended to realize the desired behavior. During training, these MPC parameters are tested in a closed-loop simulation such that behavior adaptation rewards can be computed. Further, to enhance and emphasize the generalization capabilities of the RLVR-trained LLM, the RL training is conducted on a simplistic circle map, while being evaluated on a complex racing track (see the track layouts in Appendix B).

Bubble 1: MPCxR1 – Example Prompt

Adapt the tuneable parameters of the MPC so that the car achieves the following: ***"Drive at 1.83 m/s as closely as possible"***.
This is the MPC formulation: `MPC Formulation`.
Return format:

```
new_mpc_params = {
    param1: new_value1,
    ...
}
```

Reward Modeling for Control Adaptability: To train the *MPCxR1* module, we use three different rewards, a driving, a formatting, and a parameter extraction reward, that feed into the total reward, R_{MPCxR1} as follows:

$$R_{\text{MPCxR1}} = R_{\text{drive}} + R_{\text{fmt}} + R_{\text{param}}, \quad R_{\text{drive}} = \max\left(\frac{E_{\mathcal{B}}^{\text{MPC}} - E_{\mathcal{B}}^{\text{LLM}}}{E_{\mathcal{B}}^{\text{MPC}}}, -4\right) \tag{3}$$

The aim of R_{drive}, is to reward the LLM for producing MPC parameters that result in the correct driving behavior requested by the human prompt. At each training step, these parameters are applied to the MPC and the vehicle completes a lap in simulation. Subsequently, the performance can be evaluated using behavior-specific RMSE metrics, computed for both the default MPC and the LLM adjusted MPC. We denote these errors by $E_{\mathcal{B}}^{\text{MPC}} \in [0, \infty)$ and $E_{\mathcal{B}}^{LLM} \in [0, \infty)$, respectively. The driving reward $R_{\text{drive}} \in [1, -4]$ is then computed as the relative improvment achieved by the LLM over the default MPC parameters as shown in Equation (3). Each behavioral prompt has a corresponding RMSE formulation, enabling consistent reward computation across tasks. R_{fmt} is a formatting reward similar to that in Section 3.2, encouraging the model to structure its response and reason about its answer. Finally, R_{param} is a parameter extraction reward that should discourage the LLM from producing hallucinated or invalid MPC parameters, which would result in extraction failures. This composite reward guides the LLM through GRPO to generate interpretable, valid, and behavior-aligned control adaptations.

4 Experiments and Results

4.1 Training Details

All RLVR training was done using GRPO adapted from the LoRA-based `unsloth` implementation [23] and executed on consumer-grade GPUs (RTX 4070 Ti, RTX 3090). While LoRA-based GRPO significantly reduces VRAM requirements compared to the original *DeepSeek R1* implementation, it remains significantly more computationally intensive than standard LoRA SFT training. As a result, no models larger than *Qwen2.5-3B* could be trained with GRPO, requiring approximately 11 GB of VRAM. The *Qwen2.5-7B* model, used in the baseline of [32], was limited to SFT-only training, while both 1.5B and 3B models were also SFT-trained for comparison, following the procedure in [32]. GRPO training was performed for 750 steps, resulting in an approximately 24 h-long training for the 3B model. For more details on the training procedure, refer to Appendix A.

4.2 Decision Making Results

Table 1 compares the decision-making accuracy of baseline models with SFT and RLVR finetuned models. In this comparison, all models, including *GPT-4o*, are augmented with the same RAG data (more information in Appendix E). To ensure comparability, we adopt the same evaluation procedure as in [32]. The test set comprises 8 human prompts, each querying adherence to a specific driving behavior (e.g., "Is the car driving on the racing line?"). For each behavior, the dataset includes 25 robot state histories exhibiting that behavior, resulting in a total of 200 state trajectories. By pairing each of the 8 prompts with all 200 trajectories, a comprehensive test set of 1600 prompt–trajectory combinations is constructed.

LLM	SFT	RLVR	Accuracy [%]↑
GPT4o	✗	✗	**92.48**
Qwen2.5-1.5B	✗	✗	47.09
Qwen2.5-3B	✗	✗	61.24
Qwen2.5-7B	✗	✗	82.47
Qwen2.5-1.5B	✓	✗	68.80
Qwen2.5-3B	✓	✗	83.78
Qwen2.5-7B	✓	✗	**87.32**
Qwen2.5-1.5B	✗	✓	64.17
Qwen2.5-3B	✗	✓	**78.05**
Qwen2.5-1.5B	✓	✓	82.83
Qwen2.5-3B	✓	✓	**86.82**

TABLE 1: Decision-making accuracy across LLMs sizes, illustrating the impact of SFT, and RLVR.

When finetuning models with SFT, we observe that the 1.5B, 3B, and 7B models achieve higher decision accuracies than their original counterparts, where the accuracies increase by 21.71%, 22.54%, and 4.85%-points for 1.5B, 3B, and 7B, respectively.

Due to the high memory demands of GRPO, training 7B models was not feasible on our training infrastructure (see Section 4.1). When 1.5B and 3B models are finetuned solely with RLVR,

we observe that it is less effective in improving decision-making accuracy compared to their SFT-only counterparts, achieving an improvement compared to the original base models of 17.08% and 16.81%-points for the 1.5B and 3B models.

Finally, we evaluate a two-stage training process using SFT as a pretraining step before applying RLVR. This boosts decision accuracy when compared against SFT-only trained counterparts. For *Qwen2.5-1.5B* by 14.03%-points (SFT: 68.8% / SFT+RLVR: 82.83%) and for *Qwen2.5-3B* by 3.04%-points (SFT: 83.78% / SFT+RLVR: 86.82%) over SFT-only training.

4.3 Control Adaptability Results

To quantitatively evaluate the impact of a fully embodied training procedure with closed-loop simulator interaction on the LLM, we assess control adaptability by measuring how well the robot's behavior aligns with human prompts. The agent is compared against the default MPC behavior (which is to track the racing line), allowing us to compute a control adaptability improvement score using the open-source *F1TENTH* simulator [39, 42]. SFT training, by distilling *GPT-4o* outputs on the smaller LLMs, follows the procedure outlined in [32], and all evaluations are conducted using the same RAG context to ensure comparability, see Appendix E for further details.

In contrast to the *MPCxR1* training phase, where prompts are sampled from the behavioral set $\mathcal{B}$, each behavior in $\mathcal{B}$ was rephrased and randomized five times to construct the evaluation set $\mathcal{B}'$. Additionally, the evaluation environment differs from the training setup: while *MPCxR1* was trained on a simple circular track, evaluation is conducted on a more complex and realistic track layout (see Appendix B). This setup enables testing of generalization by evaluating the model's ability to interpret a randomized behavioral set $\mathcal{B}'$ (more information in Appendix C), highlighting the open-vocabulary capabilities of the LLM, and to transfer behavior learned on a simple circular training track to a more complex, realistic environment.

LLM	SFT	RLVR	$E_C[m]\downarrow$	$E_V[ms^{-1}]\downarrow$	$E_R[ms^{-1}]\downarrow$	$E_S[ms^{-2}]\downarrow$	Ext. Fail [#]↓	Improve [%]↑
MPC (default)	-	-	0.68	1.98	5.44	1.77	-	-
GPT-4o	✗	✗	0.65 (5.0%)	**0.14 (93.2%)**	**0.12 (97.8%)**	**1.10 (38.0%)**	**0**	**58.5%**
Qwen2.5-1.5B	✗	✗	0.88 (-29.9%)†	0.90 (54.5%)†	2.55 (53.1%)	1.76 (0.6%)	3	19.6%†
Qwen2.5-3B	✗	✗	**0.62 (8.2%)**	1.51 (23.6%)	0.14 (97.5%)	1.61 (9.5%)	**0**	34.7%
Qwen2.5-7B	✗	✗	0.68 (0.3%)	0.23 (88.2%)	0.23 (95.7%)	1.43 (19.3%)	**0**	50.7%
Qwen2.5-1.5B	✓	✗	**0.49 (28.1%)**	1.01 (49.0%)	0.32 (94.2%)	4.01 (-126.1%)	**0**	11.3%
Qwen2.5-3B	✓	✗	0.67 (1.9%)†	0.43 (78.4%)	0.13 (97.6%)	1.36 (23.6%)	1	50.4%†
Qwen2.5-7B	✓	✗	0.53 (21.5%)	**0.24 (87.9%)**	**0.11 (97.9%)**	**1.34 (24.5%)**	**0**	**58.0%**
Qwen2.5-1.5B	✗	✓	0.62 (8.5%)	1.89 (4.3%)†	3.70 (32.0%)†	1.89 (-6.7%)	3	9.6%†
Qwen2.5-3B	✗	✓	**0.45 (33.3%)**	**0.39 (80.2%)**	**0.2 (96.3%)**	**1.44 (18.8%)**	**0**	**57.2%**
Qwen2.5-1.5B	✓	✓	0.62 (8.5%)	1.05 (47.0%)	0.72 (86.7%)	1.36 (23.6%)	**0**	41.5%
Qwen2.5-3B	✓	✓	**0.41 (39.9%)**	**0.19 (90.2%)**	**0.48 (91.2%)**	**1.21 (31.8%)**	**0**	**63.3%**

TABLE 2: Quantitative comparison of LLM control adaptation through MPC interactions, evaluated on *The Grand Tour* map (Appendix B). Metrics include deviation from the centerline (E_C [m]), reference velocity tracking error (E_V [ms^{-1}]), reversing error (E_R [ms^{-1}]), and driving smoothness (E_S [ms^{-2}]). MPC parameter extraction failures (Ext. Fail [#]) are where the LLM output could not be parsed due to invalid parameters. Percentage improvements are computed relative to the baseline MPC, which tracks the racing line. The improvement column aggregates overall performance across all metrics. Each entry represents the mean over five independent runs using different prompts in $\mathcal{B}'$. † indicates averages computed excluding extraction failures.

As visible from Table 2, if neither SFT nor RLVR is applied (i.e., the original models are used), control adaptability improves with model size. *GPT-4o* performs the best at 58.5%, closely followed by *Qwen2.5-7B* at 50.7%, both showing strong adaptability. However, *Qwen2.5-1.5B* fails in 3 instances due to unparseable or hallucinated MPC parameters, which is detrimental for deployment on a robot.

Among local SFT-only trained models, solely *Qwen2.5-7B* achieves a comparable improvement (58%), on par with *GPT-4o*. In contrast, *Qwen2.5-1.5B* achieves only 11.3%, and while *Qwen2.5-3B* reaches 50.4%, it suffers from parameter extraction failures, which would result in complete

disobedience of robotic behavior adaptation. Evaluating RLVR alone, the 7B model cannot be trained due to computational constraints. The 1.5B model performs worse than under SFT, with improvement dropping from 11.3% to 9.5%† and three extraction failures. Conversely, the 3B model benefits from RLVR, improving from 50.4% to 57.2%, a gain of 6.8%.

When combining SFT and RLVR, both *Qwen2.5-1.5B* and *Qwen2.5-3B* show substantial gains — 41.5% and 63.3%, respectively — with no extraction failures. Compared to SFT-only training (analogous to learning from a driving manual), the combined SFT pretraining and closed-loop RLVR training (analogous to studying the manual and taking real driving lessons) yield a 20.2%-point improvement for *Qwen2.5-1.5B* (SFT: 11.3% / SFT+RLVR: 41.5%) and a 12.9%-point improvement for *Qwen2.5-3B* (SFT: 50.4%† / SFT+RLVR: 63.3%), with SFT-only still showing critical extraction failures †, so the improvment would arguably be even higher.

These findings indicate that a 3B model can outperform even the cloud-based *GPT-4o* — despite its significantly larger parameter count — in an embodied robotic setting. While *GPT-4o* achieves a 58.5% improvement in control adaptability, our proposed training procedure reaches 63.3% with the *Qwen2.5-3B* model. These results demonstrate that LLMs can learn to interact with robots through RL, with performance gains attributed to the embodied nature of the proposed closed-loop training.

4.4 Deployment on Physical Robot

In this experiment, the proposed system is deployed on a physical 1:10 scaled open-source autonomous racing car [39] running a `Q5_k_m` quantized *Qwen2.5-3B* model on a Jetson Orin AGX OBC inferred through `llama.cpp` [43], with a full computational analysis in Appendix F. The deployed embedded model achieves a mean throughput of ∼38.78 tokens/s and an average latency of ∼8.3 s. The robot is initially placed in a perturbed state, where a human operator has manually altered the MPC parameters to induce unstable oscillations. The *MPCxR1* module is then prompted with the instruction: *"Drive smoothly at 2 m/s"*. As observed in Figure 4, the robot's behavior adapts to satisfy both objectives simultaneously, demonstrating the ability to handle compound instructions, leveraging the open-vocabulary skillset of LLMs. A second prompt, *"Reverse the car"*, followed by a final prompt, *"Drive normal again"*, are succesfully executed (not visible in Figure 4, but verifiable in Appendix D).

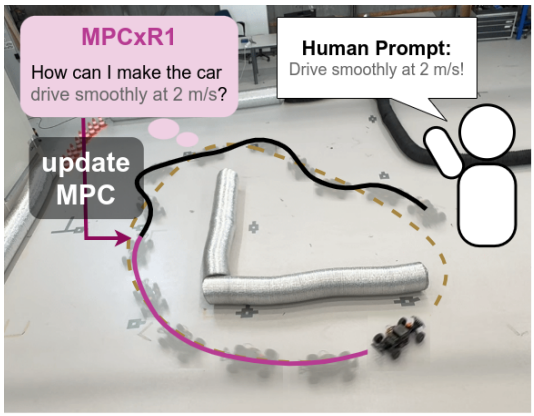

Figure 4: Adaptation of robot behavior in response to user prompts during embedded deployment with the proposed RLVR and SFT trained and `Q5_k_m` quantized *Qwen2.5-3B*.

5 Conclusion

This work presents an RL-based training procedure that enables LLMs in robotic Embodied AI settings to acquire behaviors through *"learning by doing"*, rather than relying on static datasets or solely SFT. The proposed method demonstrates that closed-loop interaction can be crucial for small models to successfully learn embodied behaviors. The approach is evaluated in an ADS setting and enables compact LLMs to adapt robotic control behavior effectively — capabilities previously limited to larger models. Using this framework, a *Qwen2.5-1.5B* model achieves up to a 20.2%-point improvement over SFT-only training, while the *Qwen2.5-3B* model reaches a 63.3% control adaptability score, surpassing the performance of the much larger cloud-based *GPT-4o* (58.5%). These results highlight the viability of feedback-driven learning for LLMs, enabling compact models in robotic Embodied AI settings through closed-loop RL.

6 Limitations

The proposed system serves as a proof of concept demonstrating that LLMs can be trained through interaction with its environment. While the results indicate that this approach is promising, several limitations remain. First, the current system does not scale well with the trend toward massive parallelization in robotic RL training [30]. In traditional RL settings, for example, with lightweight locomotion NNs, parallel robot simulation can be exploited, due to the assumption of NN inference being significantly faster than simulation, allowing the RL policy to learn from many concurrent robot observations. However, integrating LLMs into an RL loop introduces substantial computational costs, making inference longer and, as such, simulation parallelization less effective. Second, the simulation environment here is based on Robot Operating System (ROS) and tightly coupled with an MPC controller, which can become brittle during extended training runs. In particular, instability or crashes in the MPC solver can halt training, making the training process challenging. Finally, while the proposed framework is in principle applicable to other robotic systems where LLMs can interact with a high-level controller, the current implementation is tailored specifically to an autonomous car platform. Adapting it to other robotic domains would require integration work and system-specific adaptations.

References

[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. Nuscenes: A Multimodal Dataset for Autonomous Driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[2] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[3] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open X-Embodiment: Robotic Learning Datasets and Rt-X Models: Open X-Embodiment Collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[4] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

[5] C. Schuhmann, R. Vencu, R. Beaumont, R. Kaczmarczyk, C. Mullis, A. Katta, T. Coombes, J. Jitsev, and A. Komatsuzaki. LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs, 2021. URL https://arxiv.org/abs/2111.02114.

[6] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai. BEVFormer: Learning Bird's-Eye-View Representation From Multi-Camera Images Via Spatiotemporal Transformers. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IX*, page 1–18, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-20076-2. doi:10.1007/978-3-031-20077-9_1. URL https://doi.org/10.1007/978-3-031-20077-9_1.

[7] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An Open-Source Vision-Language-Action Model. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*,

volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

[8] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[9] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. S. Ryoo, G. Salazar, P. R. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. H. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:10.15607/RSS.2023.XIX.025.

[10] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, T. Unterthiner, D. Keysers, S. Koppula, F. Liu, A. Grycner, A. A. Gritsenko, N. Houlsby, M. Kumar, K. Rong, J. Eisenschlos, R. Kabra, M. Bauer, M. Bosnjak, X. Chen, M. Minderer, P. Voigtlaender, I. Bica, I. Balazevic, J. Puigcerver, P. Papalampidi, O. J. Hénaff, X. Xiong, R. Soricut, J. Harmsen, and X. Zhai. PaliGemma: A Versatile 3B VLM for Transfer. *CoRR*, abs/2407.07726, 2024. URL https://doi.org/10.48550/arXiv.2407.07726.

[11] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

[12] B. Liu, Y. Zhu, C. Gao, Y. Feng, qiang liu, Y. Zhu, and P. Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=xzEtNSuDJk.

[13] S. Zhang, Z. Xu, P. Liu, X. Yu, Y. Li, Q. Gao, Z. Fei, Z. Yin, Z. Wu, Y.-G. Jiang, and X. Qiu. VLABench: A Large-Scale Benchmark for Language-Conditioned Robotics Manipulation With Long-Horizon Reasoning Tasks, 2024. URL https://arxiv.org/abs/2412.18194.

[14] L. Wen, D. Fu, X. Li, X. Cai, T. Ma, P. Cai, M. Dou, B. Shi, L. He, and Y. Qiao. Dilu: A Knowledge-Driven Approach to Autonomous Driving With Large Language Models. *arXiv preprint arXiv:2309.16292*, 2023.

[15] M. Pavone. Decision Making and Control With LLMs. Lecture presented at NVIDIA GTC 2024, 2024. URL https://www.nvidia.com/en-us/on-demand/session/gtc24-s62855/. Accessed: 2024-07-02.

[16] J. Duan, W. Yuan, W. Pumacay, Y. R. Wang, K. Ehsani, D. Fox, and R. Krishna. Manipulate-Anything: Automating Real-World Robots Using Vision-Language Models. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 5326–5350. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/duan25a.html.

[17] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as I Can, Not as I Say: Grounding Language in Robotic Affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[18] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025.

[19] B. Zhao, Z. Wang, J. Fang, C. Gao, F. Man, J. Cui, X. Wang, X. Chen, Y. Li, and W. Zhu. Embodied-R: Collaborative Framework for Activating Embodied Spatial Reasoning in Foundation Models via Reinforcement Learning. *arXiv preprint arXiv:2504.12680*, 2025.

[20] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021.

[21] I. Petrov, J. Dekoninck, L. Baltadzhiev, M. Drencheva, K. Minchev, M. Balunović, N. Jovanović, and M. Vechev. Proof or Bluff? Evaluating LLMS on 2025 Usa Math Olympiad. *arXiv preprint arXiv:2503.21934*, 2025.

[22] Y. Yan, Y. Lu, R. Xu, and Z. Lan. Do PhD-level LLMs Truly Grasp Elementary Addition? Probing Rule Learning vs. Memorization in Large Language Models. *arXiv preprint arXiv:2504.05262*, 2025.

[23] D. Han, M. Han, and U. team. Unsloth. https://github.com/unslothai/unsloth, 2023. Accessed: April 2025.

[24] J. Pan, J. Zhang, X. Wang, L. Yuan, H. Peng, and A. Suhr. TinyZero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

[25] T. P. Swaminathan, C. Silver, and T. Akilan. Benchmarking Deep Learning Models on NVIDIA Jetson Nano for Real-Time Systems: An Empirical Investigation. *arXiv preprint arXiv:2406.17749*, 2024.

[26] J. Chen and X. Ran. Deep Learning With Edge Computing: A Review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019. doi:10.1109/JPROC.2019.2921977.

[27] Z. Lu, X. Li, D. Cai, R. Yi, F. Liu, X. Zhang, N. D. Lane, and M. Xu. Small Language Models: Survey, Measurements, and Insights. *arXiv preprint arXiv:2409.15790*, 2024.

[28] D. Silver and R. S. Sutton. Welcome to the Era of Experience. *DeepMind*, 2025.

[29] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024.

[30] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 91–100. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/rudin22a.html.

[31] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, et al. Outracing Champion Gran Turismo Drivers With Deep Reinforcement Learning. *Nature*, 602(7896):223–228, 2022.

[32] N. Baumann, C. Hu, P. Sivasothilingam, H. Qin, L. Xie, M. Magno, and L. Benini. Enhancing Autonomous Driving Systems With on-Board Deployed Large Language Models, 2025. URL https://arxiv.org/abs/2504.11514.

[33] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, et al. Language to Rewards for Robotic Skill Synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

[34] S. Ismail, A. Arbues, R. Cotterell, R. Zurbrügg, and C. A. Alonso. NARRATE: Versatile Language Architecture for Optimal Control in Robotics. *arXiv preprint arXiv:2403.10762*, 2024.

[35] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-Level Reward Design via Coding Large Language Models. *arXiv preprint arXiv:2310.12931*, 2023.

[36] A. Dao and D. B. Vu. AlphaMaze: Enhancing Large Language Models' Spatial Intelligence via GRPO. *arXiv preprint arXiv:2502.14669*, 2025.

[37] A. Azzolini, H. Brandon, P. Chattopadhyay, H. Chen, J. Chu, Y. Cui, J. Diamond, Y. Ding, F. Ferroni, R. Govindaraju, et al. Cosmos-Reason1: From Physical Common Sense to Embodied Reasoning. *arXiv preprint arXiv:2503.15558*, 2025.

[38] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, et al. Qwen Technical Report. *arXiv preprint arXiv:2309.16609*, 2023.

[39] N. Baumann, E. Ghignone, J. Kühne, N. Bastuck, J. Becker, N. Imholz, T. Kränzlin, T. Y. Lim, M. Lötscher, L. Schwarzenbach, et al. ForzaETH Race Stack—Scaled Autonomous Head-to-Head Racing on Fully Commercial Off-the-Shelf Hardware. *Journal of Field Robotics*, 2024.

[40] Y. Zhai, H. Bai, Z. Lin, J. Pan, S. Tong, Y. Zhou, A. Suhr, S. Xie, Y. LeCun, Y. Ma, and S. Levine. Fine-Tuning Large Vision-Language Models as Decision-Making Agents via Reinforcement Learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=nBjmMF2IZU.

[41] A. Szot, M. Schwarzer, H. Agrawal, B. Mazoure, R. Metcalf, W. Talbott, N. Mackraz, R. D. Hjelm, and A. T. Toshev. Large Language Models as Generalizable Policies for Embodied Tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=u6imHU4Ebu.

[42] V. S. Babu and M. Behl. F1tenth. Dev-an Open-Source Ros Based F1/10 Autonomous Racing Simulator. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1614–1620. IEEE, 2020.

[43] G. Gerganov and O.-S. Contributors. Llama.cpp. https://github.com/ggerganov/llama.cpp, 2023. Accessed: April 2025.

Appendix

A Training Curves

This section presents the RLVR training curves for both the *MPCxR1* and *DecisionxR1* modules, shown in Figure 5 and Figure 6, respectively. The models used in both training setups are *Qwen2.5-1.5B* and *Qwen2.5-3B*, each pretrained via SFT. The training runs correspond to the final two rows in Table 2 and Table 1, where performance is evaluated after both SFT and RLVR training.

The left plots illustrate the total reward (R_{MPCxR1} and $R_{\text{DecisionxR1}}$) over training steps, showing consistent reward maximization in both cases. In *MPCxR1*, the reward emerges from closed-loop interactions with the simulator, whereas in *DecisionxR1*, it is learned from static dataset supervision.

The right plots show the average output token length over training. Interestingly, in both cases, the LLM learns to produce shorter, more concise outputs, despite the fact that brevity is not explicitly incentivized through the reward function. This contrasts with the behavior observed in *DeepSeek R1* [18], where longer Chain of Thoughts (CoTs) were rewarded for multi-step mathematical reasoning. These results suggest that the reasoning tasks explored in this work differ from mathematical settings, and align with recent findings in spatial reasoning with VLMs trained using RLVR on static datasets [19].

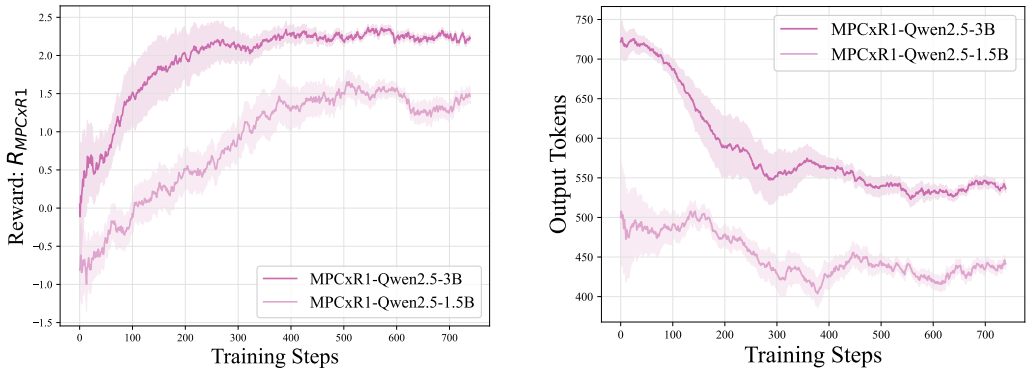

Figure 5: Visualization of *MPCxR1* RLVR (GRPO) training with the standard deviation shaded. *Qwen2.5* 3B and 1.5B LLMs, both pretrained via SFT, are used as base models. Left: The reward signal R_{MPCxR1} that the LLM learns to maximize through interaction with the simulation environment. Right: The average output token length.

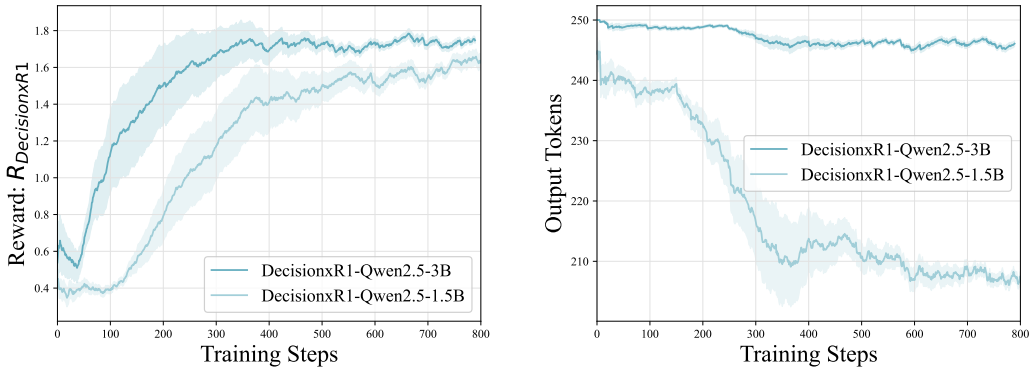

Figure 6: *DecisionxR1* RLVR (GRPO) training curves. Left: Reward $R_{\text{DecisionxR1}}$ learned from static binary adherence classification. Right: Output token length over the training steps.

B MPCxR1 Training and Evaluation Map

The maps used in the proposed *MPCxR1* RLVR training and evaluation procedure are shown in Figure 7. The **Circle** map, used during training, provides a minimal and structured environment in which the LLM can efficiently learn to interact with the robot and environment. In contrast, **The Grand Tour** map, used for control adaptability evaluation as in Table 2, follows a conventional racing circuit design with greater complexity and variation.

This deliberate separation between training and evaluation environments is intended to assess the generalization capabilities of the learned driving behaviors. By training on a simple layout and testing on a more challenging and realistic track, the system's ability to transfer learned control adaptations across domains is effectively demonstrated.

C Behavioral Sets in Training and Evaluation

To give more insight into the behavioral set used during RLVR training of *MPCxR1*, we provide the list of behaviors the LLM is trained to adapt toward. These behaviors form the training set $\mathcal{B}$ and are evaluated on the simple **Circle** track. Importantly, the reference RMSE values ($E_{\mathcal{B}}^{\text{MPC}}$) presented below correspond specifically to this training map. As such, they differ from the values reported in Table 2, where evaluation was conducted on **The Grand Tour** map, a more complex and realistic evaluation track.

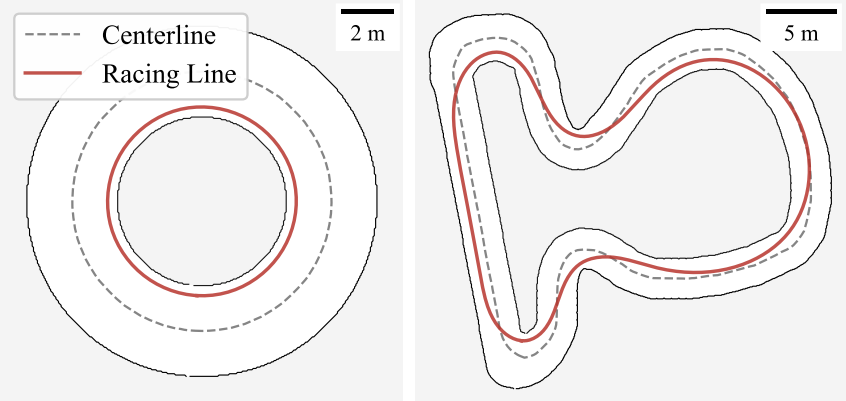

Figure 7: Tracks used for training and evaluation. Left: **Circle** map used for *MPCxR1* RLVR training. Right: **The Grand Tour** map used for evaluation. Dashed grey and solid red lines denote the centerline and the minimum-curvature racing line (targeted by the default MPC), respectively.

$\mathcal{B}$ during training on the **Circle map** consists of:

I **Centerline tracking:** Prompt: *"Drive on the centerline"*. The RMSE evaluation E_C computes the distance toward the centerline over a lap. The default MPC tracks a minimum-curvature racing line, deviating from the centerline, yielding $E_C^{\text{MPC}} = 1.23$.

II **Velocity adherence:** Prompt: *"Drive at 1.83 m/s as closely as possible"*. The RMSE evaluation E_V computes the deviation from the reference velocity of $1.83\,\mathrm{m\,s^{-1}}$. As the default MPC aims for time-optimal speed, this yields $E_V^{\text{MPC}} = 2.1$.

III **Reversing:** Prompt: *"Reverse the car"*. The RMSE evaluation E_R computes the deviation from $-1\,\mathrm{m\,s^{-1}}$ (negative velocity indicating reverse motion), where $E_R^{\text{MPC}} = 4.93$.

IV **Smooth driving:** Prompt: *"Drive smoothly"*. The RMSE evaluation E_S measures deviation from $0\,\mathrm{m\,s^{-2}}$ acceleration. Due to the uniform curvature of the circular track, the default MPC exhibits low acceleration, resulting in $E_S^{\text{MPC}} = 0.38$.

The randomized and perturbed behavioral set $\mathcal{B}'$ during evaluation, which are unseen during SFT and RLVR training, on the **Grand Tour map** consists of. Here, the mean of each $E_{\mathcal{B}'}^{\text{MPC}}$ corresponds to the MPC default parameters in Table 2:

i **Centerline tracking:** Prompts: [*"Stay directly on the middle of the track"*, *"Follow the track by staying aligned with the middle of the track"*, *"Drive away as far as possible from the walls"*, *"Ensure that the distance to the left and right wall remain the same"*, *"Drive on the centerline"*]. The RMSE evaluation E_C computes the distance toward the centerline over a lap. The default MPC tracks a minimum-curvature racing line, deviating from the centerline, yielding $E_C^{\text{MPC}} = 0.679$, which is the same for all 5 prompts.

ii **Velocity adherence:** Prompts: [*"Set the driving speed to 3.5 m/s"*, *"Target a driving speed of 2.2 meters per second"*, *"Move at a constant speed of 1.25 m/s"*, *"Travel at 2.9 meters per second"*, *"Adjsut the speed to exactly 4.5 m/s"*]. The RMSE evaluation E_V computes the deviation from the reference velocity of [3.5, 2.2, 2.25, 2.9, 4.5] $\mathrm{m\,s^{-1}}$ respectively. As the default MPC always aims for the time-optimal speeds with respect to a range of reference velocities, this yields $E_V^{\text{MPC}} = [1.362, 2.417, 3.301, 1.808, 1.045]$ respectively.

iii **Reversing:** Prompts: [*"Slowly back the vehicle up"*, *"Reverse the vehicle"*, *"Switch to reverse and drive backwards"*, *"Retreat by reversing the car"*, *"Drive the car backwards"*]. The RMSE evaluation E_R computes the deviation from $-1\,\mathrm{m\,s^{-1}}$ (negative velocity indicating reverse motion), where $E_R^{\text{MPC}} = 5.439$, which is the same for all 5 prompts.

iv **Smooth driving:** Prompts: [*"Drive in a fluid and controlled manner"*, *"Maintain low jerk and high smoothness in driving behavior"*, *"Focus on smooth driving"*, *"Drive smoothly"*, *Reduce*

lateral acceleration"]. The RMSE evaluation E_S measures deviation from $0\,\mathrm{m\,s^{-2}}$ acceleration. The default MPC exhibits rather high acceleration levels due to the non-uniform track layout, resulting in $E_S^{\mathrm{MPC}} = 1.773$.

D Qualitative LLM Outputs

This is the chat log of the experiment conducted on the physical robot in Figure 4. Initially, the robot has been manually configured to oscillate (by lowering the heading trackign cost `qalpha` to 0, and by over inflating the `track_safety_margin` to $1\,\mathrm{m}$ on a narrow and edgy track), then *MPCxR1* is tasked to drive smoothly at $2\,\mathrm{m\,s^{-1}}$, followed by a command to reverse the car and then to drive nominally again. The prompt bubble's are omitted after the first one for visibility reasons.

Note that the computation times within this experiment might differ from the results obtained in Appendix F, as the LLM inference was now performed with the entire autonomy stack running in parallel, which is computationally expensive.

Bubble 2a: MPCxR1 – Instruction

"Drive smoothly at 2m/s"

Bubble 2b: MPCxR1 – Prompt

Adapt the tuneable parameters of the MPC so that the car achieves the following: ***"Drive smoothly at 2m/s"***.
This is the MPC formulation: `MPC Formulation`.
Return format:

```
new_mpc_params = {
    param1: new_value1,
    param2: new_value2,
    ...
}
```

Bubble 2c: MPCxR1 – Response

```
new_mpc_params = {'qv': 10, 'qn': 20, 'qalpha': 7, 'qac':
0.01, 'qddelta': 0.1, 'v_max': 2.0, 'v_min': 1.0}
```

Human Narration

The robot was initially oscillating heavily, as in Figure 4, due to the manually induced behavior of setting `qalpha` to zero and by inflating overly `track_safety_margin` to $1\,\mathrm{m}$. The *MPCxR1* correctly limits the velocity to $2\,\mathrm{m\,s^{-1}}$, and adapts `qalpha` to 7. The robot is now driving smoothly at $2\,\mathrm{m\,s^{-1}}$ on the racing line. **Computation time** on the Jetson Orin AGX: $9.12\,\mathrm{s}$.

Bubble 2d: MPCxR1 – Instruction

"Reverse the car"

Bubble 2e: MPCxR1 – Response

```
new_mpc_params = {v_max: -1, v_min: -2}
```

Human Narration

The robot was tracking the racing line, when prompted to reverse. The *MPCxR1* module correctly identified the relevant parameters that enforce reversing behavior, by only allowing the MPC to operate within negative velocity constraints. The robot is now reversing. **Computation time** on the Jetson Orin AGX: 5.84 s.

Bubble 2f: MPCxR1 – Instruction

"Drive normal again"

Bubble 2g: MPCxR1 – Response

```
new_mpc_params = {'qv': 10, 'qn': 20, 'qalpha': 7, 'qac':
0.01, 'qddelta': 0.1, 'alat_max': 10, 'a_min': -5, 'a_max':
5, 'v_min': 1, 'v_max': 5, 'track_safety_margin': 0.45}.
```

Human Narration

The robot is now driving normally again by using the default MPC parameter values. **Computation time** on the Jetson Orin AGX: 9.93 s

E Additional RAG Information

This section provides supplementary details on the RAG information used in both the *DecisionxR1* and *MPCxR1* modules. The design and usage closely follow the implementation presented in [32], Section III-B. This appendix is intended to offer additional insight into the structure and purpose of the retrieved context used to support reasoning and adaptation during training and inference.

The RAGs of Listing 1 and Listing 2, are meant to augment the prompts for their respective *MPCxR1* and *DecisionxR1* modules, respectively. This is a simple way in which a human operator can quickly integrate new system knowledge into the LLM. For example, if one were to deploy the proposed system on a full-scale car, changing the nominal speeds in Listing 2 higher than those observed on a 1:10 scaled racing car, would be as simple as editing `Hint 2` in Listing 2 through the RAG `.txt` file.

E.1 RAG for Controller Adaptation

```
# Memory Entry 0:
Scenario:
To force going forwards v_min should be positive. If you want it to be able to reverse, then
    set v_min to negative.
MPC Action:
mpc_params = {
    'v_min': positive, if you want to go forwards, else negative to reverse
}

# Memory Entry 1:
Scenario:
Always have v_max be higher than v_min.
MPC Action:
mpc_params = {
    'v_max': higher than v_min
}

# Memory Entry 2:
Scenario:
To drive far away from the walls, the boundary inflation should be increased. That way it will
     drive closer to the centerline. This is the most important to stay away from the walls.
MPC Action:
mpc_params = {
```

```
    'track_safety_margin': high
}

# Memory Entry 3:
Scenario:
To reverse the car slowly on the racing line, the v_min must be negative!
MP Action:
mpc_params = {
    'v_min': negative,
}
...
```

Listing 1: RAG Memory for *MPCxR1*. There are 11 memories in total.

E.2 RAG for Decision-Making

```
# Hint 1:
If the d-speed is above than 0.5m/s is high.

# Hint 2:
Unless specified differently by the human, the car is usually driving at speeds between 5 and
    7m/s.

# Hint 3:
If the distance to a wall is smaller than 0.4m, the car is close to that wall. Staying close
    to the wall means maintaining a consistent distance below 0.4m.

# Hint 4:
If the s-speed is close to 0m/s, then the car is stopped.

# Hint 5:
The car is reversing if the s-speed is negative or if the s-coordinate is reducing over
    multiple samples.
...
```

Listing 2: RAG Memory for *DecisionxR1*. There are 11 hints in total.

F Computation

This section analyzes the computational characteristics of the proposed training and deployment pipeline, focusing primarily on model quantification and its impact on both performance and deployability on embedded hardware. To assess the trade-offs between reasoning capability and computational efficiency in real-world robotic applications, we evaluated different model sizes, precision levels, and deployment environments.

To reduce inference latency, we quantized the SFT and RLVR trained LLMs by merging the LoRA adapters into the base model and applying post-training quantization, which converts the models to a `Q5_k_m` quantized model, denoted as Q5, and exported in `GGUF` format and run using the `llama.cpp` inference engine [43]. Full-precision models are here denoted as FP16, while quantized variants are referred to as Q5.

To quantify the inference speedup gained by quantization and the optimized inference engine [27, 43], we measured token throughput, inference latency, and memory usage of each model using the same input prompt. Each measurement was performed 10 times sequentially and conducted on two hardware platforms: an RTX 3090 GPU, representing high-end consumer hardware, and a Jetson Orin AGX, targeting onboard deployment.

The results are summarized in Table 3 and show that quantized models offer considerable gains in throughput, especially on embedded hardware. On the Jetson AGX, for instance, the Q5 1.5B model achieves a latency of 1.82 s and a throughput of 56.14 tokens/s, compared to the FP16 version's 14.88 s latency and 6.99 tokens/s. This corresponds to a 7.5x reduction in latency, making real-time inference feasible on embedded hardware. Similar trends are also observed in the other model sizes. Note that latency is determined through the token throughput and the number of output tokens produced, which is not the same for each model.

HW	LLM	Quant	Param [#B]	Mem [GB]	Tokens [#]	Tokens/s [s^{-1}]↑	Latency [s] μ_t
RTX 3090	Qwen	FP16	1.5	2.7	104	**33.88**	**3.07**
			3	4.4	109	18.03	6.05
			7	8.4	108	8.7	12.41
	Qwen	Q5	1.5	1.7	102	**203.89**	**0.50**
			3	2.8	137	141.57	0.97
			7	5.6	40	100.74	0.40
Jetson Orin AGX	Qwen	FP16	1.5	2.2	104	**6.99**	**14.88**
			3	3.8	109	3.55	30.73
			7	6.9	108	1.60	67.45
	Qwen	Q5	1.5	1.6	102	**56.14**	1.82
			3	2.7	145	38.78	3.74
			7	5.3	40	23.36	**1.71**

TABLE 3: Comparison of computational performance of quantized and full-precision models. The LLMs were deployed on both an RTX 3090 GPU and the *Jetson Orin AGX* robotic OBC. The number of tokens denotes the output tokens generated for the given inference. The average inference latency is denoted with μ_t.

We further evaluated the impact of post-training quantization on the reasoning capabilities of our newly trained models, with a particular focus on decision-making and control adaptability. Table 4 displays how quantization affects decision accuracy across different model sizes. While full-precision models (FP16) consistently outperform their quantized counterparts, the accuracy drop due to Q5 quantization remains moderate. The largest decrease in accuracy is observed for the *Qwen2.5-3B* model, where performance drops from 86.82% to 79.88%. Nevertheless, the quantized SFT & RLVR-trained model still outperforms the base model and other training configurations. This suggests that the reasoning capabilities learned through RLVR remain largely intact in quantized models. As a result, the quantization of RLVR-trained models leads to meaningful gains in speed and memory efficiency with only minor reductions in decision accuracy, making real-time onboard inference feasible.

LLM	Quant	SFT	GRPO	Accuracy [%]↑
Qwen2.5-7B	FP16	✓	✗	**87.32**
Qwen2.5-1.5B	FP16	✓	✓	82.83
Qwen2.5-3B	FP16	✓	✓	**86.82**
Qwen2.5-7B	Q5	✓	✗	**86.95**
Qwen2.5-1.5B	Q5	✓	✓	79.63
Qwen2.5-3B	Q5	✓	✓	**79.88**

TABLE 4: Impact of quantization on decision-making accuracy across multiple LLM sizes and training configurations.

Table 5 highlights that control adaptation, unlike decision making, places greater demands on the model, requiring it to generate valid, structured parameters that directly influence low-level control. This complexity makes the task more sensitive to reasoning stability, particularly under quantization.

The quantized 1.5B model, while achieving a solid improvement of 49.1%, exhibits multiple extraction failures. These errors suggest that its reasoning capacity is insufficient to meet the stricter requirements of closed-loop control consistently. In contrast, the *Qwen2.5-3B* Q5 model delivers more reliable performance, achieving 47.9% improvement without any failures across all prompts.

This robustness under quantization makes the 3B model a better candidate for embedded deployment, as it strikes a balance between computational efficiency and control adaptability without sacrificing correctness.

LLM	Quant	SFT	GRPO	$E_C[m] \downarrow$	$E_V[ms^{-1}] \downarrow$	$E_R[ms^{-1}] \downarrow$	$E_S[ms^{-2}] \downarrow$	Ext. Fail [#]↓	Improve [%]↑
MPC (default)	-	-	-	0.68	1.98	5.44	1.77	-	-
Qwen2.5-7B	FP16	✓	✗	0.53 (21.5%)	0.24 (87.9%)	**0.11 (97.9%)**	1.34 (24.5%)	**0**	58.0%
Qwen2.5-1.5B	FP16	✓	✓	0.62 (8.5%)	1.05 (47.0%)	0.72 (86.7%)	1.36 (23.6%)	**0**	41.5%
Qwen2.5-3B	FP16	✓	✓	**0.41 (39.9%)**	**0.19 (90.2%)**	0.48 (91.2%)	**1.21 (31.8%)**	**0**	**63.3%**
Qwen2.5-7B	Q5	✓	✗	**0.57 (16.3%)**	0.90 (54.6%)	**0.12 (97.8%)**	1.48 (16.6%)	**0**	46.4%
Qwen2.5-1.5B	Q5	✓	✓	0.59 (13.4%)	**0.65 (67.2%)**	1.18 (78.3%)†	1.10 (37.9%)†	3	**49.1%**†
Qwen2.5-3B	Q5	✓	✓	0.65 (3.7%)	0.93 (53.1%)	0.32 (94.1%)	**1.05 (40.8%)**	**0**	47.9%

TABLE 5: Control adaptability evaluation of `Q5_k_m` quantized models (Q5) compared to their full-precision (FP16) counterparts. Metrics include deviation from the centerline (E_C [m]), reference velocity tracking error (E_V [ms^{-1}]), reversing accuracy (E_R [ms^{-1}]), and driving smoothness (E_S [ms^{-2}]). MPC parameter extraction failures (Ext. Fail [#]), are where the LLM output could not be parsed due to invalid parameters. Percentage improvements are computed relative to the baseline MPC, which tracks the racing line. The improvement column aggregates overall performance across all metrics. Each entry represents the mean over five independent runs using different prompts. † indicates averages computed excluding extraction failures.