White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Sheffield.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/246973/

Version: Accepted Version

Proceedings Paper:
Wang, Y., Vyas, J., Liu, T. et al. (2026) Closed-loop control with rule-aligned small language models and multi-agent self-correction. In: 2026 IEEE Conference on Control Technology and Applications (CCTA). 2026 IEEE Conference on Control Technology and Applications (CCTA), 12-14 Aug 2026, Vancouver, BC, Canada. Institute of Electrical and Electronics Engineers (IEEE), pp. 52-57. ISBN: 9798319531476. ISSN: 2768-0762. EISSN: 2768-0770.

https://doi.org/10.1109/ccta62090.2026.11684321

© 2026 The Authors. Except as otherwise noted, this author-accepted version of a conference paper published in 2026 IEEE Conference on Control Technology and Applications (CCTA) is made available via the University of Sheffield Research Publications and Copyright Policy under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

UNIVERSITY OF LEEDS
University of Sheffield
UNIVERSITY of York

Closed-Loop Control with Rule-Aligned Small Language Models and Multi-Agent Self-Correction

Yuchen Wang
School of Computer Science
University of Sheffield
Sheffield, UK
ywang1016@sheffield.ac.uk

Javal Vyas
Department of Chemical Engineering
Imperial College London
London, UK
j.vyas24@imperial.ac.uk

Tong Liu
School of Computer Science
University of Sheffield
Sheffield, UK
t.liu@sheffield.ac.uk

Mehmet Mercangöz
Department of Chemical Engineering
Imperial College London
London, UK
m.mercangoz@imperial.ac.uk

arXiv:2607.09713v1 [cs.AI] 24 Jun 2026

***Abstract*— A key step toward autonomous industrial operation is the ability to create and reconfigure control policies from natural-language requirement specifications, with minimal or no manual redesign. In this setting, policy generation by AI agents can be a credible path when paired with a plant-aware validator (e.g., a digital twin) that can check generated candidate actions before execution. However, practical deployment is constrained by inference latency and compute footprint: large cloud-based models are often too slow, opaque, or data-sensitive for edge closed-loop use. This work investigates whether a compact Small Language Model (SLM) can be retrained for control reasoning and embedded in a validator-guided correction loop. We use a Qwen2.5-1.5B model aligned via Group Relative Policy Optimization (GRPO), combined with (i) an action agent, (ii) a symbolic/digital-twin-style validation layer, and (iii) a reprompting agent that iteratively steers outputs toward valid actions. In randomized thermal-control simulations (30 experiments with 500 steps each), the framework achieves 91.5% average action-alignment accuracy (86.3%–100% across cases) at 3.84 s mean inference latency. Under symbolic re-mapping, it maintains a 95% in-range rate, indicating robust physical regulation despite reduced token-level agreement. These results support SLM+validator architectures as a practical path toward reconfigurable autonomous control at the edge.**

Keywords: Autonomous systems, Small language models, Edge AI, Generative AI

I. Introduction

A central requirement for autonomous industrial operation is the ability to *create and reconfigure control policies* from high-level, natural-language requirement specifications with minimal manual redesign. This is particularly important in environments where objectives and constraints evolve over time (e.g., energy priorities, safety envelopes, and operating policies). In this setting, policy generation by AI agents can be a credible path when paired with a plant-aware validator (e.g., a digital twin) that can check generated candidate actions before execution [1].

Conventional industrial control architectures are highly effective for deterministic numerical regulation, but they are less flexible when symbolic operating intent must be incorporated quickly. Translating new expert directives into deployed logic typically requires manual re-engineering of rules, supervisory layers, or interlocks, which slows reconfiguration and increases engineering overhead. This creates a persistent gap between high-level operational intent and low-level control execution.

Language models provide a promising interface for closing this gap because they can parse and operationalize natural-language instructions and support iterative self-correction during inference [2], [3]. However, deploying cloud-hosted Large Language Models (LLMs) in closed-loop industrial workflows remains challenging: API dependence introduces variable end-to-end latency, while external data transfer raises governance and sovereignty concerns for sensitive operational telemetry [4], [5]. These issues are especially limiting for edge and offline autonomy.

This work investigates whether a compact Small Language Model (SLM) can be trained to perform control-oriented reasoning and embedded into a validator-guided correction loop. We develop a local architecture built around Qwen2.5-1.5B [6], where policy behavior is aligned using Group Relative Policy Optimization (GRPO) [7]. A key advantage of this RL-based alignment is that the model autonomously generates its own chains-of-thought (CoT). While a brief warm-up distillation is employed, it serves strictly for format alignment rather than task-specific memorization. Furthermore, the framework operates in a zero-shot capacity regarding corrective interactions; although GRPO is utilized solely to enhance the agent's numerical and logical reasoning, the Action Agent is never exposed to the diverse reprompting hints during training. Nevertheless, it demonstrates an emergent ability to dynamically update its decisions based on such corrective feedback. The controller is implemented as three cooperating components: (i) an Action Agent that proposes control moves, (ii) a symbolic/digital-twin-style validator that checks action validity, and (iii) a Reprompt Agent that provides corrective guidance when validation fails. This design targets a practical balance between reasoning capability, latency, and edge deployability.

Contributions: The main contributions are threefold: (1) *Rule-aligned SLM policy learning*, providing a pipeline that aligns a 1.5B SLM with thermal control logic using GRPO; (2) a *zero-shot validator-guided inference loop* that enables compact SLMs to exhibit emergent correction capabilities without explicit corrective training; and (3) an *empirical evaluation* demonstrating strong physical regulation (95% IRR) under interface perturbations in randomized simulations.

II. Related Work

A. LLM-driven Autonomous Control and its Advantages

Traditional automation systems primarily rely on rigid, hard-coded logic to manage industrial processes. While these systems are highly reliable for numerical regulation, they struggle to integrate complex, symbolic expert intents. Recent research has shifted toward LLM-based Autonomous Agents, where models act as "reasoning engines" to bridge this gap. This paradigm shift has demonstrated significant potential in laboratory discovery [8], digital workflows [9], and robotic reward engineering [10].

B. Enhancing Numerical Reasoning in Language Models

Thermal management requires precise numerical comparison and threshold-based logic, areas where standard LLMs often struggle with "stochastic drift" or calculation errors [11]. Current strategies to mitigate this include:

Reinforcement Learning (RL): Techniques like PPO and particularly Group Relative Policy Optimization (GRPO) [7], [12] have emerged to align model outputs with verifiable rewards, reinforcing the logical consistency behind numerical decisions.

Self-Correction and Recursive Reasoning: To stabilize control outputs, recent research employs external validation loops and reprompting agents [13], [14]. However, purely external designs often lack the intrinsic logical depth required for real-time stability. Frameworks like the **Self-Taught Reasoner (STaR)** [15] address this by enabling models to iteratively refine their own reasoning chains (**Chain-of-Thought**) based on successful outcomes. This allows the model to internalize complex logic during training—an evolution we extend by using GRPO to foster the "Aha-moment" [7] within small-scale architectures, ensuring they reason correctly without the overhead of heavy external supervision.

Tool Utilization: Although agents can call external calculators [16], [17], the resulting communication latency and instability make this approach not suitable for all problems.

C. Small Language Models (SLMs) for Edge Deployment

While Large Language Models (LLMs) possess superior reasoning, their prohibitive inference latency and deployment costs hinder real-time industrial integration. Small Language Models (SLMs), when properly optimized, are being explored to achieve functional performance in specialized tasks while offering the potential for local execution in thermal control [18]. Recent studies emphasize using SLMs as "control-specific" agents that internalize expert experience to replace massive general-purpose architectures [12]. This paper extends this direction by investigating the feasibility of addressing the reasoning gap in 1.5B-parameter SLMs through GRPO-based heuristic internalization.

Algorithm 1 Multi-Agent Collaborative Control Loop

Input: Current system state s_t, Maximum retry limit N_{max}
Output: Validated control action u_t
$attempt \leftarrow 0$
$hint \leftarrow \emptyset$
while $attempt < N_{max}$ **do**
  $u_t, \text{CoT}_t \leftarrow \text{ActionAgent}(s_t, hint)$
  $u_{gt} \leftarrow \text{ExpertRules}(s_t)$
  if $u_t = u_{gt}$ **then**

    return u_t
  else
    $hint \leftarrow \text{RepromptAgent}(u_t, u_{gt})$
    $attempt \leftarrow attempt + 1$
  end if
end while

return u_{gt}

III. Methodology

Our architecture employs a multi-agent collaboration to ensure system's self-correction.

A. Collaborative Multi-Agent Architecture

The core of the closed-loop system consists of three specialized agents that partition the tasks of decision-making, verification, and logical correction.

Action Agent (Reasoning Core): Powered by Qwen2.5-1.5B and aligned via GRPO, this agent processes the state s_t and context to generate a control action u_t alongside a Chain-of-Thought (CoT) reasoning path.

Validation Agent (Symbolic Guardrail): Enforcing a strict logical filter, this component flags the output u_t as *UNSAFE* if it diverges from the ground-truth expert rule u_{gt}.

Reprompt Agent (Logical Interpreter): Also utilizing Qwen2.5-1.5B, this agent diagnoses discrepancies between u_t and u_{gt}. Fine-tuned on the Action Agent's error distributions, it constructs a semantic *Corrective Hint* h_t regarding boundary or rule oversights to guide zero-shot self-correction.

B. The Control Algorithm

The interaction between these agents is formalized in Algorithm 1 and visualized in the system architecture shown in Fig. 1. As illustrated, the framework is structured into three functionally distinct layers: the *Reasoning Layer* (blue), the *Symbolic Validation Layer* (orange), and the *Execution Layer* (green).

C. Control Logic and System Design

The symbolic rules embedded within the framework are engineered to emulate an expert thermal engineer's heuristics, focusing on three operational pillars:

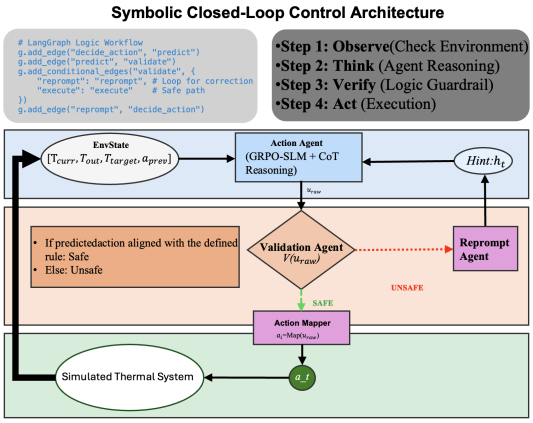


Fig. 1. Schematic of the Agentic Closed-Loop Control Architecture. The framework incorporates a GRPO-aligned Small Language Model (SLM) for decision reasoning (blue) and a Symbolic Validation (orange) for rule checking. The Execution layer(green) is to update the system continuously after receiving one valid action.

*1) **Control Logic Design**:* The symbolic rules in Table I are engineered to emulate the decision-making process of a thermal system. Unlike reactive systems, our framework incorporates **temporal state persistence**: the agent's current decision is conditioned not only on the thermal state S but also on the **memory of the previous action** (A_{prev}). This "Latching" mechanism (P3) ensures mission continuity and prevents oscillatory switching during thermal transitions, maintaining stability until the setpoint is reached.

TABLE I

SYMBOLIC RULE HIERARCHY FOR THERMAL CONTROL

Priority	**Condition (Current State S)**	**Action**
1. Safety Override (Critical, P1)		
Safety H	$T_{cur} \geq T_{safe,high}$	$-2Q$
Safety L	$T_{cur} \leq T_{safe,low}$	$+2Q$
2. Boundary Trigger (Tracking, P2)		
Upper B	$T_{cur} > T_{target,high}$	$-Q$
Lower B	$T_{cur} < T_{target,low}$	$+Q$
3. Mission Latching (Hysteresis, P3)		
Maint. H	$T_{cur} \in [T_{low}, T_{high}] \cap A_{prev} > 0$	$+Q$
Maint. C	$T_{cur} \in [T_{low}, T_{high}] \cap A_{prev} < 0$	$-Q$
Default	All other applicable cases	0

*2) **Closed-Loop Reasoning and Validation**:* The control loop is organised as a *verify-then-execute* pipeline (Algorithm 1). At each step, a candidate action is first proposed by the reasoning agent and then screened by a validation layer before it is applied to the plant.

3) System Dynamics and Environment: The thermal environment is modeled via a first-order energy balance equation:

$$C\frac{dT_t}{dt} = UA(T_{\text{out}} - T_t) + a_t + \epsilon \tag{1}$$

where C is the room thermal capacity, UA is the overall heat-transfer coefficient, and T_{out} is the ambient outdoor temperature. The control action is selected from the discrete set $a_t \in \{-2Q, -Q, 0, Q, 2Q\}$, where $Q = 5.0$ is the base power unit.

D. ALGORITHM REFINEMENT: GRPO AND DISTILLATION

To optimize the SLM's reasoning capabilities, we employ a strategy of Group Relative Policy Optimization (GRPO) followed by Knowledge Distillation.

1) GRPO and Reward Function: Unlike PPO which requires a value function $V(s)$, we adopt GRPO to stabilize training via group-based relative advantages [7]. For a given state s_t, we sample G reasoning trajectories $\{y_1, y_2, \ldots, y_G\}$. The reward function R is formulated to align the model with symbolic logic. For each trajectory i, a reward r_i is assigned based on the logic detailed in Table II.

TABLE II

REWARD LOGIC AND PENALTY MAPPING FOR GRPO TRAINING

Condition	**Ground truth**	**Output**	**Score**
Task persistence (latching)			
$a_{t-1} \in \{+Q, +2Q\}$[a]	$+Q$	0 (early exit)	-0.5
$a_{t-1} \in \{-Q, -2Q\}$[a]	$-Q$	0 (early exit)	-0.5
Critical errors			
Polarity inversion	e.g., $+Q$	e.g., $-Q$	-0.75
Invalid format	Any	Unknown	-1.0
Correct execution	a^*	$\hat{a} = a^*$	1.0

The optimization follows the GRPO algorithm [19]. For each state s_t, the SLM generates a group of $G = 8$ independent reasoning trajectories.

2) Action Agent Initialization via Distillation: The overall training and alignment workflow is illustrated in Fig. 2. To prevent divergence during the subsequent reinforcement learning phase, the **Action Agent** undergoes an initial brief distillation stage using *DeepSeek-R1-671B* as the teacher. Following the *distilling step-by-step* paradigm [20], the agent learns to internalize the structured reasoning paths (CoT) alongside the control labels using a subset of 5,000 trajectories. This stage provides a necessary "warm-start" policy, ensuring that the agent maintains a structured format before being optimized for numerical precision via GRPO.

3) Iterative Error-Driven Distillation: Inspired by the *STaR* (Self-Taught Reasoner) framework [15], we develop an iterative distillation pipeline for the **Reprompt Agent**, forming the feedback loop shown in the right panel of Fig. 2. The process is executed in three stages:

1) Error Profiling: After the Action Agent completes its GRPO optimization, we deploy it in challenging thermal environments to collect corner-case failure trajectories. **2) Teacher Correction:** These error cases are fed into the DeepSeek-R1-671B teacher model to generate 5,000 corrective samples. **3) Aha-Moment Distillation:** Drawing inspiration from the "Aha moment," the Reprompt Agent is fine-tuned to initiate feedback with a "Wait..." prefix, forcing a secondary verification.

E. Prompt Design for Symbolic Reasoning

The prompt structures are developed independently for the Action and Reprompt agents. As illustrated in Fig. 3, the Action Agent is provided with a structural template focusing

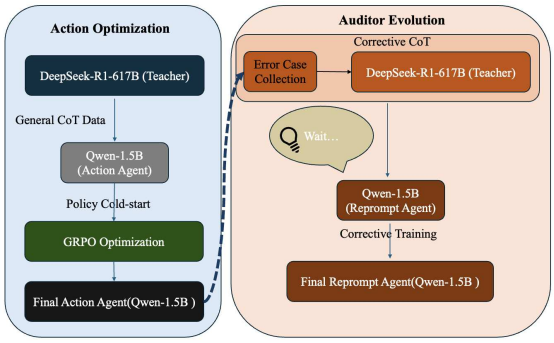


Fig. 2. The bifurcated training and self-correction pipeline, featuring a GRPO-based action optimization path (left) and an iterative auditor evolution path (right).

on state interpretation, while the Reprompt Agent prompt in Fig.4 is equipped with a "Logic Auditor" role. Specifically, at any given State t, the auditor's prompt is designed to provide hints on the chain-of-thoughts the action agent results.

Example 3.2: Action Agent Prompt Design

```
You are a precision thermal controller. Select exactly ONE action: [+2Q, +Q, 0, -Q, -2Q].
MATCH ONLY THE FIRST APPLICABLE RULE FROM TOP TO BOTTOM.
### 1. SAFETY OVERRIDE (Critical)
- IF Current >= Safety Upper: Answer: -2Q
- IF Current <= Safety Lower: Answer: +2Q
### 2. BOUNDARY TRIGGER (Out of Range)
- IF Current > Target Upper: Answer: -Q
- IF Current < Target Lower: Answer: +Q
### 3. MISSION LATCHING (Within Target Range [Target Lower, Target Upper])
- IF Previous Action was (+Q or +2Q):
- IF Current <= Target Value: Answer: +Q
- IF Previous Action was (-Q or -2Q):
- IF Current >= Target Value: Answer: -Q
- IF Previous Action was 0: Answer: 0
### DEFAULT:
- In all other cases: Answer: 0
### STRICT REQUIREMENTS:
- Use EXACTLY the numerical values provided in the Data.
- Once a rule matches, STOP. Do not provide multiple actions.
```

Reprompt Block

Validation Feedback:
- **IF** a_{t-1} violates symbolic rules.
  - **Hint:** "Wait... Current temperature is outside safety limits. Adjust required."
- **ELSE:** Proceed to execution.

```
**Output Format:**
1. Reasoning: Compare Current with thresholds carefully.
2. Answer: <Action_Label>
```

User Prompt (Example State t)

```
Data:
Target Value: 22.0, Current: 22.9
Target Range: [21.5, 22.5]
Safety Range: [18.0, 25.0]
Previous Action: +Q
```

Fig. 3. Hierarchical Prompt Design with Symbolic Latching and Reprompt Logic. The structured instruction set ensures deterministic decision-making through prioritized rule matching.

IV. Experimental Results and Analysis

We report three evaluation protocols: (i) randomized closed-loop robustness over 30 experiments with 500 steps each (15,000 total steps), (ii) cross-model comparison on a separate 1,500-sample benchmark, and (iii) symbolic re-mapping stress tests for interface robustness. Metrics are reported within each protocol and are not mixed across protocols.

A. Experimental Setup and Baselines

The thermal system is simulated with physical parameters $C = 20.0$ and $UA = 0.5$, using a base power unit $Q = 5.0$.

We employ three distinct baselines: (1) **General-Purpose Large Language Models (LLMs)**, including GPT-4o and DeepSeek-V3 as high-capacity neural baselines; (2) an **Ablation Baseline (Base-SLM)**, utilizing the original Qwen2.5-1.5B without GRPO; and (3) a **Heuristic Rule-based Controller (HRBC)**, representing the upper bound of purely symbolic consistency.

Example: Reprompt Agent Prompt Design

```
You are a Logic Auditor. You will be shown a temperature control scenario and an Action Agent's
INCORRECT reasoning.\n\n
"TASK:\n"
"1. Analyze the numerical contradiction between the Data and the Agent's Thought.\n"
"2. Generate a one-sentence corrective hint to guide the Agent.\n\n"
"STRICT OUTPUT FORMAT:\n"
"\nWait... [Your one-sentence hint starting with 'Wait...']\n</Hint>\n\n"
"CONSTRAINTS:\n"
"- The Hint must NOT contain any numbers or correct category names.\n"
"- The Hint must focus on the boundary that was missed or the comparison that was wrong.
```

User Prompt (Example State t)

```
Data:
Target Value: 22.0, Current: 22.9
Target Range: [21.5, 22.5]
Safety Range: [18.0, 25.0]
ACTION AGENT'S WRONG REASONING:[Chain-of-Thoughts]
Previous Action: +Q
```

Fig. 4. Example of the Reprompt Agent's prompt design and a specific user prompt at State t. The Logic Auditor provides a corrective hint without revealing the direct answer, forcing the Action Agent to re-reason based on numerical boundaries.

B. Performance on Randomized Scenarios

To evaluate the control robustness and generalization capability of the proposed framework, we conducted simulations encompassing 30 randomized test cases, totaling 15,000 operational steps.

A key objective of these tests is to assess the agent's robustness. The randomized parameters include temperature values and disturbance profiles that were **explicitly excluded from the training set**, thereby testing the model's ability to generalize its learned control logic to unseen scenarios. The robust performance is verified across 30 randomized trials (15,000 steps total) with unseen initial states $T_0 \in [0, 45]\,^\circ\text{C}$, setpoints $T_{\text{set}} \in [0, 40]\,^\circ\text{C}$, and dynamic ambient disturbances $T_{\text{out}} \in [-20, 40]\,^\circ\text{C}$.

We evaluate performance using two metrics: (1) **Action Alignment Accuracy (Acc)**, which measures the logical consistency between the predicted action u_t and expert ground truth u_{gt}, defined as $Acc = \frac{1}{N}\sum_{t=1}^{N}\mathbb{I}(u_t = u_{gt})$ (where $\mathbb{I}(\cdot)$ is the indicator function); and (2) **Inference Latency (Lat)**, evaluating real-time feasibility by averaging the computational time per step: $Lat = \frac{1}{N}\sum_{t=1}^{N}(t_t^{\text{end}} - t_t^{\text{start}})$, where t_t^{start} and t_t^{end} denote the timestamps at the beginning and completion of inference for step t, respectively.

*1) **Effectiveness of Training**:* To evaluate the impact of the training process, we compared the performance of the proposed agent against the original Qwen2.5-1.5B model. Both models were tested within the same multi-agent framework to ensure a fair assessment.

As illustrated in Table III, the base model achieved only 39.5% accuracy, struggling to comprehend the specific thermal control constraints. After training, the SLM's accuracy increased to 91.5%, representing a significant performance leap. This result clearly demonstrates that the proposed optimization methods enable the SLM to execute reasoning tasks that were previously beyond the its capabilities. **Comparative Benchmarking (SLM vs. LLMs)** Table IV presents the performance metrics across 1,500 samples. While DeepSeek-V3 achieves the highest accuracy (98.47%), the proposed SLM maintains a highly competitive 96.67% fidelity while reducing inference latency by 38.9% compared to DeepSeek-V3 and 14.7% compared to GPT-4o.

*2) **Comparative Benchmarking (SLM vs. LLMs)**:* Table IV presents the performance metrics across 1,500 samples. While DeepSeek-V3 achieves the highest accuracy (98.47%),

TABLE III

PERFORMANCE COMPARISON: BASE MODEL VS. FINE-TUNED SLM (BOTH INCLUDE REPROMPTING PROCESS)

Model Configuration	Acc. ↑	Total Samples
Qwen2.5-1.5B (Original)	39.5%	2,000
Ours (Fine-tuned SLM)	**91.5%**	**15,000**

the proposed SLM maintains a highly competitive 96.67% fidelity while reducing inference latency by 38.9% compared to DeepSeek-V3 and 14.7% compared to GPT-4o. It is

TABLE IV

COMPARATIVE PERFORMANCE ACROSS 1,500 SAMPLES

Model	Acc.	Latency (Lat)	Avg. Reprompts
GPT-4o	93.07%	4.50s	0.16
DeepSeek-V3	**98.47%**	6.28s	**0.09**
Ours (SLM)	96.67%	**3.84s**	0.38

noteworthy that the SLM exhibits a higher average reprompt rate (0.38) compared to the LLMs. This indicates that the SLM relies more heavily on the iterative validation mechanism to correct initial reasoning. However, even with these additional correction cycles, the SLM's total latency remains the lowest. This result suggests a clear optimization path: by further enhancing the SLM's zero-shot reasoning precision through advanced distillation, the frequency of reprompting can be minimized, which would lead to a further reduction in operational latency.

*3) **Case Study: Performance Analysis of a Typical Scenario**:* As shown in 5, we analyze a representative scenario with an initial temperature of 42°C and a target setpoint of 25°C. This case serves to illustrate the SLM agent's **execution fidelity** relative to the expert-defined HRBC baseline.

In 6, the SLM agent (black line) demonstrates a temperature trajectory that is **nearly identical to the HRBC baseline** (red dash-dotted line). While **GPT-4o** may exhibit high individual action accuracy, its resulting thermal trajectories often display **irregular fluctuations or sub-optimal convergence slopes** (as seen in the red lines). In contrast, our SLM agent (black line) maintains a trajectory that is **consistent** with the HRBC baseline.

The SLM agent's behavior remains **strictly bounded** within the target safety zone without any observed overshoot, even across an extended horizon of 500 operational steps. This demonstrates that the system can obtain long-term stability. To further evaluate the controller's adaptability, Figures 7 and 8 illustrate the system's performance across a broad spectrum of initial thermal states, ranging from cold starts (12°C) to high-heat conditions (42°C), covering the complete temperature ranges. Specifically, Figure 7 characterizes the system response in cold environments, while Figure 8 focuses on performance under hot environment temperatures.

The results demonstrate **consistent convergence** regardless of the initial starting temperature. In all scenarios, the system effectively stabilizes within the **target safety zone** ($25°\mathbf{C} \pm 2°\mathbf{C}$) within 100 operational steps. Throughout this duration, the system exhibits minimal steady-state oscillations, maintaining stable temperature regulation by strictly adhering to the defined switching logic between active and inactive states.

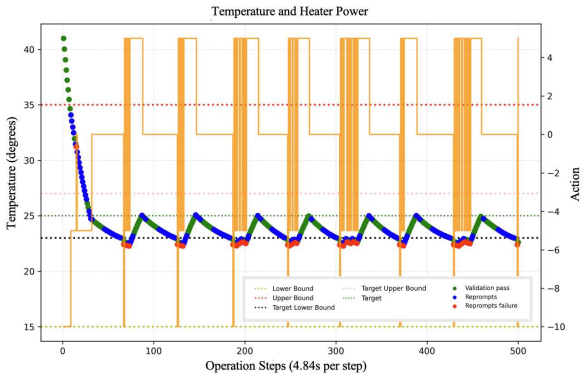


Fig. 5. Execution trace of the SLM agent. Colored markers indicate the real-time interaction between the Action Agent and the Validation Agent (Green: Pass, Blue: Reprompt Success, Red: Failure).

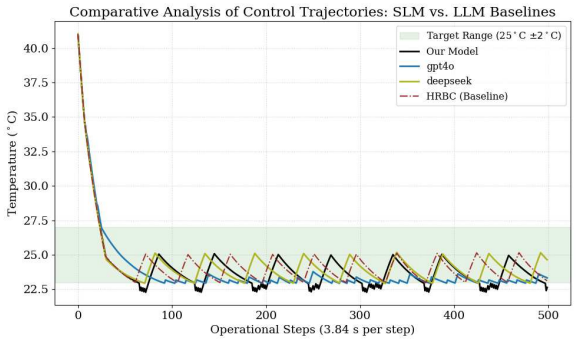


Fig. 6. Comparative analysis of control trajectories. The SLM agent (blue) exhibits high executive fidelity, closely tracking the HRBC expert baseline compared to cloud-based LLMs. Operational steps are executed at an average latency of 3.84 s.

C. Symbolic Generalization and Robustness

To evaluate the system's operational consistency, we conducted a **Symbolic Re-mapping** stress test where the heating power label Q was re-indexed to an arbitrary label F. This experiment investigates whether the model's reasoning is grounded in the underlying task logic or merely sensitive to specific symbolic strings.

In this re-mapped scenario, the SLM maintained **100% format consistency**, strictly adopting the new symbolic prefix (F) in all outputs. While the exact action alignment accuracy decreased to 74.1%, the system achieved an **In-Range Rate (IRR) of 95%**(defined here as the percentage of operational steps where the temperature remains within the prescribed target range). This disparity between symbolic accuracy (74.1%) and physical success (95%) is highly significant; it indicates that even when the agent fails to select the "expert-identical" token, its decisions remain functionally effective for maintaining thermal stability.

V. CONCLUSION

This paper investigated whether a compact, locally deployed Small Language Model (SLM) can support closed-loop thermal control within a validator-guided correction architecture. We aligned a Qwen2.5-1.5B model using Group Relative Policy Optimization (GRPO) and combined it with

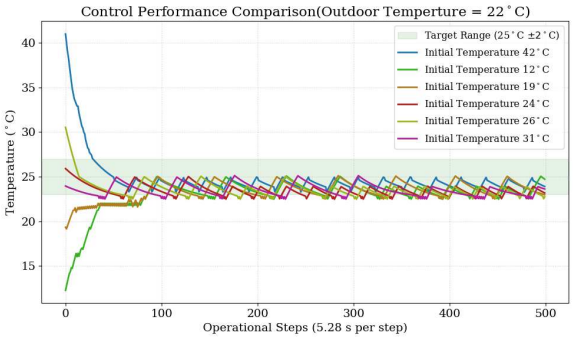


Fig. 7. Thermal control trajectories across diverse initial conditions. The shaded green area represents the target range ($25°\text{C} \pm 2°\text{C}$). Each operational step represents an average latency of $5.28\,\text{s}$. Under a cold environment ($T_{out} = 22°\text{C}$), the system demonstrates consistent convergence.

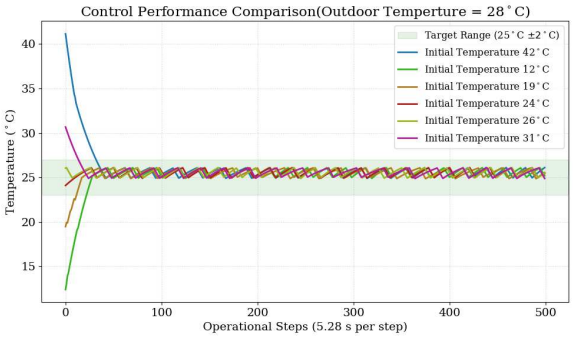


Fig. 8. Thermal control trajectories across diverse initial conditions. The shaded green area represents the target range ($25°\text{C} \pm 2°\text{C}$). Each operational step represents an average latency of $5.28\,\text{s}$. Under a hot condition ($T_{out} = 28°\text{C}$), the system demonstrates consistent convergence.

a validation layer and reprompting mechanism to iteratively steer decisions toward valid actions.

In simulation, the proposed framework achieved 91.5% average action-alignment accuracy (86.3%–100% across 30 experiments of 500 steps each), with a mean per-step inference latency of 3.84 s in the reported benchmark. In a symbolic re-mapping stress test, the controller maintained a 95% in-range rate (IRR), indicating that physical regulation remained strong even when token-level agreement decreased. Overall, these results support the feasibility of SLM+validator loops for edge-oriented, reconfigurable control workflows, while avoiding reliance on cloud-hosted inference.

A. Limitations and Future Work

While promising, several limitations guide our future directions. 1) *physical validation* via hardware-in-the-loop (HIL) testing is required to assess performance under sensor noise and actuator non-idealities. 2) we aim to enhance *validator fidelity* by replacing the symbolic rule layer with a predictive digital twin. 3) comprehensive *latency characterization* must profile jitter, variance, and tail latency on specialized embedded hardware rather than just mean latency. 4) we will extend the framework beyond single-zone thermal benchmarks to multi-variable settings to evaluate the *scalability* of the multi-agent control architecture.

REFERENCES

[1] M. S. Gill, J. Vyas, A. Markaj, F. Gehlhoff, and M. Mercangöz, "Leveraging llm agents and digital twins for fault handling in process plants," in *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2025, pp. 1–8.

[2] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.

[3] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[5] V. Pedreira, D. Barros, and P. Pinto, "A review of attacks, vulnerabilities, and defenses in industry 4.0 with new challenges on data sovereignty ahead," *Sensors*, vol. 21, no. 15, p. 5189, 2021.

[6] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Lu *et al.*, "Qwen2. 5-coder technical report," *arXiv preprint arXiv:2409.12186*, 2024.

[7] 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.

[8] M. Baldea, A. T. Georgiou, B. Gopaluni, M. Mercangöz, C. C. Pantelides, K. Sheth, V. M. Zavala, and C. Georgakis, "From automated to autonomous process operations," *Computers & Chemical Engineering*, vol. 196, p. 109064, 2025.

[9] T. L. Khoo, T. S. Lee, S.-T. Bee, C. Ma, and Y.-Y. Zhang, "A comparative review of large language models in engineering with emphasis on chemical engineering applications," *Processes*, vol. 13, no. 9, 2025. [Online]. Available: https://www.mdpi.com/2227-9717/13/9/2680

[10] 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.

[11] 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.

[12] X. Guo, D. Keivan, U. Syed, L. Qin, H. Zhang, G. Dullerud, P. Seiler, and B. Hu, "Controlagent: Automating control system design via novel integration of llm agents and domain expertise," *arXiv preprint arXiv:2410.19811*, 2024.

[13] J. Vyas and M. Mercangöz, "Autonomous industrial control using an agentic framework with large language models," *IFAC-PapersOnLine*, vol. 59, no. 6, pp. 349–354, 2025, 14th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems DYCOPS 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896325005300

[14] J. Vyas and M. Mercangoz, "Autonomous control leveraging llms: An agentic framework for next-generation industrial automation," *arXiv preprint arXiv:2507.07115*, 2025.

[15] E. Zelikman, Y. Wu, J. Mu, and N. Goodman, "Star: Bootstrapping reasoning with reasoning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 15 476–15 488, 2022.

[16] M. L. Romero and R. Suyama, "Agentic ai for intent-based industrial automation," *arXiv preprint arXiv:2506.04980*, 2025.

[17] Z. Hu, M. Li, H. Tang, and Z. Wang, "Autocontrol: An end-to-end fully automated workflow for control design of building energy systems," *Energy*, p. 138329, 2025.

[18] P. Belcak, G. Heinrich, S. Diao, Y. Fu, X. Dong, S. Muralidharan, Y. C. Lin, and P. Molchanov, "Small language models are the future of agentic ai," *arXiv preprint arXiv:2506.02153*, 2025.

[19] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan *et al.*, "Deepseek-v3 technical report," *arXiv preprint arXiv:2412.19437*, 2024.

[20] C.-Y. Hsieh, C.-L. Li, C.-K. Yeh, H. Nakhost, Y. Fujii, A. Ratner, R. Krishna, C.-Y. Lee, and T. Pfister, "Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 8003–8017.