Beyond Latency: Benchmarking Real-Time Strategic Control for LLM Agents in Evolving Environments

Anonymous authors
Paper under double-blind review

Abstract

Large language models are increasingly used in Agent tasks that require continuous observation, reasoning, and action. In real-time environments, however, the world continues to evolve during model inference. Each decision therefore spans both wall-clock time and environment time, affecting the decision frequency achievable within a fixed horizon. We develop an OpenRA-based benchmark for real-time strategic control, comprising controlled Fixed-Force diagnostics and complete Full-Skirmish games. The benchmark jointly records wall-clock response time, observation-to-action gap, decision frequency, and control outcomes. Experiments show that added waiting widens the gap while reducing decision frequency. The same wall-clock waiting time corresponds to substantially different gaps under different world speeds; the mean gap at the fastest speed is roughly three times that at the slowest. Full-Skirmish experiments further reveal clear differences in computational cost and long-horizon control performance across observation representations and control architectures. The benchmark also supports evaluation across model backbones, Bot behaviors, and opponent difficulties in continuously running RTS environments.

1 Introduction

Large language models are increasingly used in interactive tasks that require sustained observation, planning, and action. In these settings, reasoning itself takes time, yet the environment does not wait for the model to finish. As the world continues to evolve during inference, the positions, resources, or targets relevant to an action may have changed by the time that action is executed, even if it was appropriate when the observation was obtained. Recent Agent benchmarks have extended this form of persistent interaction to web operation, embodied environments, open worlds, and long-horizon games. (Liu et al., 2024; Cobbe et al., 2020; Küttler et al., 2020; Fan et al., 2022; Paglieri et al., 2025; Ma et al., 2024; Li et al., 2025)

Recent studies show that model response time can substantially affect Agent performance in dynamic and real-time environments, with several works examining settings that continue to advance during model inference. (Zhou et al., 2024; Wen et al., 2026; Kang et al., 2025; Zhang et al., 2025) Real-time control unfolds across both wall-clock time and environment time: inference consumes real time as the game world progresses. The number of decisions that can be completed within a fixed horizon consequently changes as well. Moreover, the same wall-clock response time may correspond to different numbers of environment ticks under different environment speeds. We therefore jointly record wall-clock response time, observation-to-action gap, decision frequency, and final control outcomes to analyze the relationships among these temporal quantities.

We develop an OpenRA-based benchmark for real-time strategic control. OpenRA provides a continuously running RTS world with controllable initial states and world speed; the benchmark augments this substrate by recording the environment times associated with observation generation and action application. Fixed-Force provides a controlled setting for real-time combat, whereas Full-Skirmish retains the full economy, production, technology, and long-horizon combat process. By varying the added wait before action application to-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

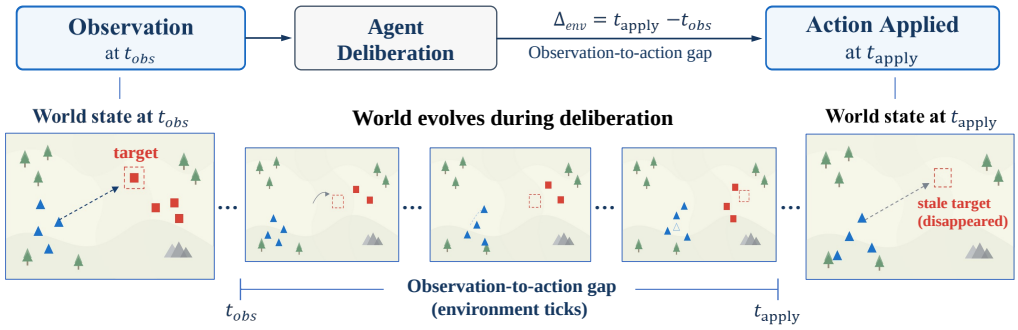


Figure 1: Observation-to-action gap in a continuously evolving environment. The Agent receives an observation at t_{obs}, and the corresponding action is accepted by the game at t_{apply}; the benchmark records the environment ticks elapsed between the two.

gether with world speed, we measure how wall-clock response time maps to the observation-to-action gap. We further analyze the accompanying changes in decision frequency and control outcomes. The benchmark also supports comparisons across observation representations, control architectures, model backbones, Bot behaviors, and opponent difficulties.

Our main contributions are:

- We introduce an OpenRA-based real-time RTS benchmark with a unified interface spanning controlled Fixed-Force diagnostics and complete Full-Skirmish games, supporting real-time combat analysis and long-horizon strategic-control evaluation, respectively.
- We provide a temporal evaluation protocol that jointly records wall-clock response time, observation-to-action gap, and decision frequency. Controlled variations in added waiting and world speed are used to study the mapping between wall-clock time and environment time.
- We evaluate observation representations, control architectures, model backbones, Bot behaviors, and opponent difficulties within a unified benchmark, together with their corresponding computational costs and long-horizon control outcomes.

2 Benchmark for Real-Time Strategic Control

We develop and package a real-time strategic-control benchmark based on OpenRA Red Alert, allowing Agents to observe, reason, and act through a unified interface in a continuously evolving RTS world. Unit states, resources, production, and combat continue to change during inference, whereas OpenRA provides controlled experimental settings for the initial state, world speed, and opponent configuration. The benchmark separates Agent-side configuration from environment-side configuration: the former supports interchangeable model backbones, observation representations, and control architectures; the latter covers task selection, added waiting before action application, world speed, Bot behavior, and opponent difficulty.

The benchmark includes two complementary tasks. Fixed-Force provides controlled diagnostics for real-time combat, whereas Full-Skirmish evaluates complete long-horizon RTS control. The two tasks respectively support controlled mechanism-level experiments and complete long-horizon games under the same environment interface.

2.1 Tasks and Evaluation

Fixed-Force diagnostics. Fixed-Force begins from predefined forces and deployments for both sides, with each episode running to a fixed horizon H. By removing long-horizon economy and production while providing full-information observations, the task focuses evaluation on temporal effects and control performance in real-time combat.

Because each episode is evaluated at a fixed horizon, we use the relative terminal ArmyValue advantage as the primary metric:

$$S_{\text{adv}} = \frac{V_{\text{own}} - V_{\text{opp}}}{V_{\text{own}} + V_{\text{opp}} + \epsilon}, \tag{1}$$

where V_{own} and V_{opp} denote the remaining ArmyValue of the two sides. S_{adv} measures the normalized terminal force advantage; the absolute terminal difference is denoted by $\Delta V = V_{\text{own}} - V_{\text{opp}}$.

Full-Skirmish games. Full-Skirmish starts from a standard RTS opening and retains resource gathering, base construction, unit production, technology progression, fog of war, and long-horizon combat. Each game continues until a native match result is produced or the benchmark horizon is reached. Win/Draw/Loss serves as the primary outcome. We additionally report the terminal ArmyValue differential, $\Delta V = V_{\text{own}} - V_{\text{opp}}$, which measures the difference in remaining force value between the two sides and distinguishes different degrees of terminal advantage or disadvantage under the same discrete outcome. If no native winner is produced before the benchmark horizon, the episode is recorded as a Draw.

2.2 Temporal Configuration

For a decision based on an observation, let t_{obs} denote the environment time at which the Agent receives that observation, and let t_{apply} denote the environment time at which the corresponding action is accepted by the game. The elapsed environment time is defined as

$$\Delta_{\text{env}} = t_{\text{apply}} - t_{\text{obs}}. \tag{2}$$

Wall-clock response time measures the real time required for the Agent to complete a decision. In contrast, Δ_{env} measures the number of environment ticks elapsed from observation generation to action application.

The benchmark can insert an additional wait d before action application or vary the world speed s. Added waiting changes the real time elapsed before action application, whereas world speed changes the number of environment ticks advanced per unit of wall-clock time. Both therefore affect the observation-to-action gap.

2.3 Agent Configuration

The benchmark constructs two observation representations from the same current game state visible to the player. Raw observation directly serializes the state of each individual entity. Structured observation retains global economic, military, and production information, while dynamically grouping some visible entities according to spatial proximity so that large numbers of individual entities need not be enumerated separately.

For control architecture, we compare Flat and Hierarchical. Flat uses a single language-model process to produce executable actions directly at each decision cycle. Hierarchical separates lower-frequency long-horizon strategic reasoning from higher-frequency production, tactical control, and concrete action execution, allowing different levels to update at different temporal scales.

3 Experiments

We first study the temporal process of real-time control, focusing on how added waiting changes the observation-to-action gap and decision frequency, as well as how world speed

alters the mapping from wall-clock time to environment time. We then turn to complete games to examine the relationships among input cost, model computation, and long-horizon control outcomes across different observation representations and control architectures. Finally, we vary the model backbone, Bot behavior, and opponent difficulty to examine how control behavior changes across different model and opponent settings.

3.1 Experimental Setup

All temporal experiments use the GLM-5.3 Flat-Structured Agent. We first insert additional waits of $0, 2, 5, 8$ s before action application to observe the joint changes in waiting time, observation-to-action gap, and decision frequency, running each setting on 8 seeds. We next test $0, 2, 5$ s of added waiting under the slowest, default, and fastest world speeds. Each combination uses the same 6 seeds, allowing us to compare the observation-to-action gaps induced by the same waiting time at different environment speeds. Finally, using 8 paired seeds, we compare two action-timing settings under a fixed decision schedule to examine gap differences at near-matched action frequency.

For the Full-Skirmish system comparison, Flat-Raw, Flat-Structured, and Hierarchical are evaluated on the same 8 seeds. The breadth experiments use Hierarchical throughout, with the same 4 seeds for every condition. Model-backbone comparisons fix the Normal Bot and Default difficulty; Bot-behavior comparisons fix GLM-5.3 and Default difficulty; difficulty comparisons fix GLM-5.3 and the Normal Bot. All remaining environment and match settings are held constant.

3.2 Temporal Effects

Figure 2(a) shows how added waiting changes the real-time control process. As the wait increases from 0 s to 8 s, the mean observation-to-action gap rises from 109 ticks to 247 ticks, whereas the mean number of decisions per episode falls from 11.6 to 4.9. Added waiting therefore changes two temporal quantities simultaneously: more environment time elapses between observation generation and action application, while fewer decisions can be completed within the fixed horizon.

Terminal combat outcomes follow a different pattern. As shown in Table 1, the 2 s setting increases the mean gap to 131 ticks and reduces the mean number of decisions to 9.3; nevertheless, the number of episodes that clear all enemy units rises from 6/8 to 7/8, with the mean number of surviving enemy tanks falling from 0.75 to 0.25. At 5 s and 8 s, cleared episodes drop to 3/8 and 1/8, respectively, while the corresponding numbers of surviving enemy tanks increase to 2.50 and 2.00. At smaller waits, the clearest changes appear in the gap and decision frequency; as waiting increases further, terminal combat progress also changes substantially. The sweep thus captures both the temporal changes within the control process and the terminal outcomes that emerge after those changes accumulate across multiple actions.

Table 1: Temporal characteristics and terminal combat outcomes under different added waits. Cleared denotes the number of episodes, out of 8, in which all enemy units are eliminated before the horizon; Own tanks and Enemy tanks report the mean numbers of surviving tanks at termination.

Delay (s)	Gap (ticks)	Decisions	Cleared	Own tanks	Enemy tanks
0	109	11.63	6/8	8.13	0.75
2	131	9.25	7/8	8.63	0.25
5	222	5.75	3/8	9.25	2.50
8	247	4.88	1/8	9.38	2.00

Figure 2(b) examines how world speed changes the mapping between wall-clock time and environment time. Under the same 2 s added wait, the slowest, default, and fastest settings produce gaps of approximately 70, 114, and 210 ticks, respectively. Across all three waiting

times, the mean gap at the fastest speed is about three times that at the slowest speed. World speed therefore determines the environment-time exposure associated with a given wall-clock waiting time: even when real-time waiting is unchanged, the amount of environment progression can differ substantially.

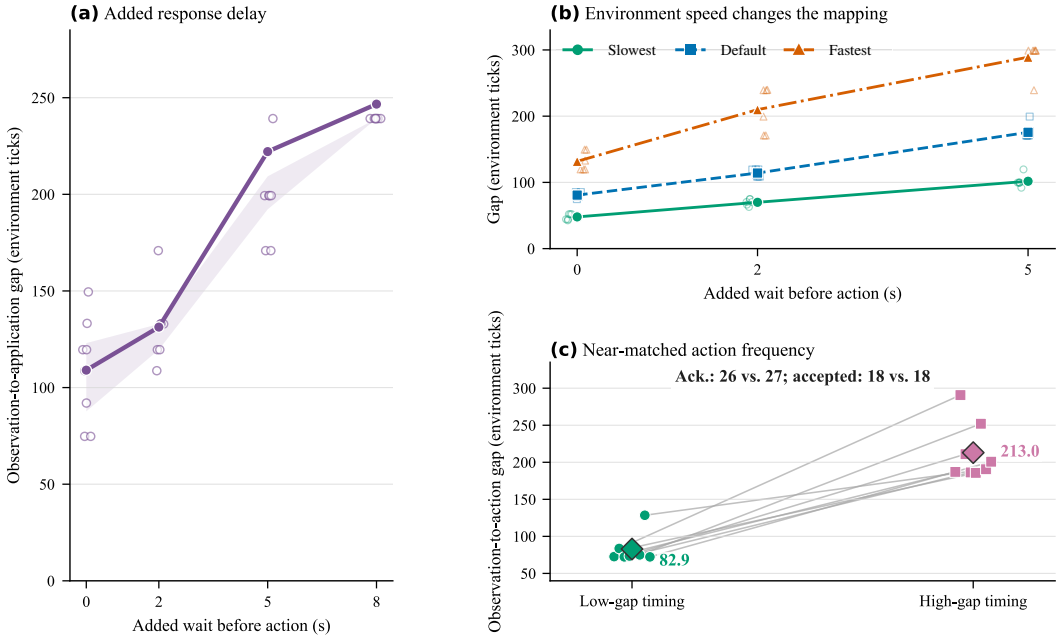


Figure 2: Temporal effects in a continuously evolving environment. (a) At the default world speed, increasing the pre-action wait enlarges the mean observation-to-action gap (IQR shown) while reducing the number of completed decisions. (b) The same wall-clock waiting time produces substantially different observation-to-action gaps at different world speeds. (c) Under a fixed decision schedule, near-matched action frequency is accompanied by clearly separated observation-to-action gaps.

Figure 2(c) examines the observation-to-action gap under near-matched decision frequency. Across 8 paired seeds, the two conditions produce 26 and 27 acknowledged action cycles, with 18 accepted action cycles in each condition. Their mean gaps are 82.9 and 213.0 ticks; the High-gap setting produces the larger gap in all 8 paired seeds. Together with the preceding experiments, these results characterize the temporal process along two related dimensions: how often the Agent receives a new decision opportunity and how much environment progression has elapsed since the observation underlying an action application.

3.3 System Design

The temporal experiments hold the Agent system fixed; we now turn to the system design itself. We first examine how observation representation changes the input processed at each decision. In Figure 3(a), with the Flat architecture held fixed, Structured reduces the mean serialized observation length from approximately 10.8k to 6.3k characters, a 42.1% reduction. Input tokens per model call decrease by 31.0%. Structured representation therefore substantially reduces both the per-decision input size and the model input cost.

This input compression is accompanied by a change in terminal battlefield state. As shown in Figure 3(b), Flat-Structured achieves a higher terminal ΔV than Flat-Raw on 6/8 paired seeds, with an average increase of approximately 3.4k. Both Flat systems record 0/8 wins; nevertheless, Structured reaches a higher terminal ΔV in most paired games while using fewer input tokens. The comparison links representation efficiency to long-horizon game state: the compressed representation reduces input cost and corresponds to a higher terminal ΔV in most paired games.

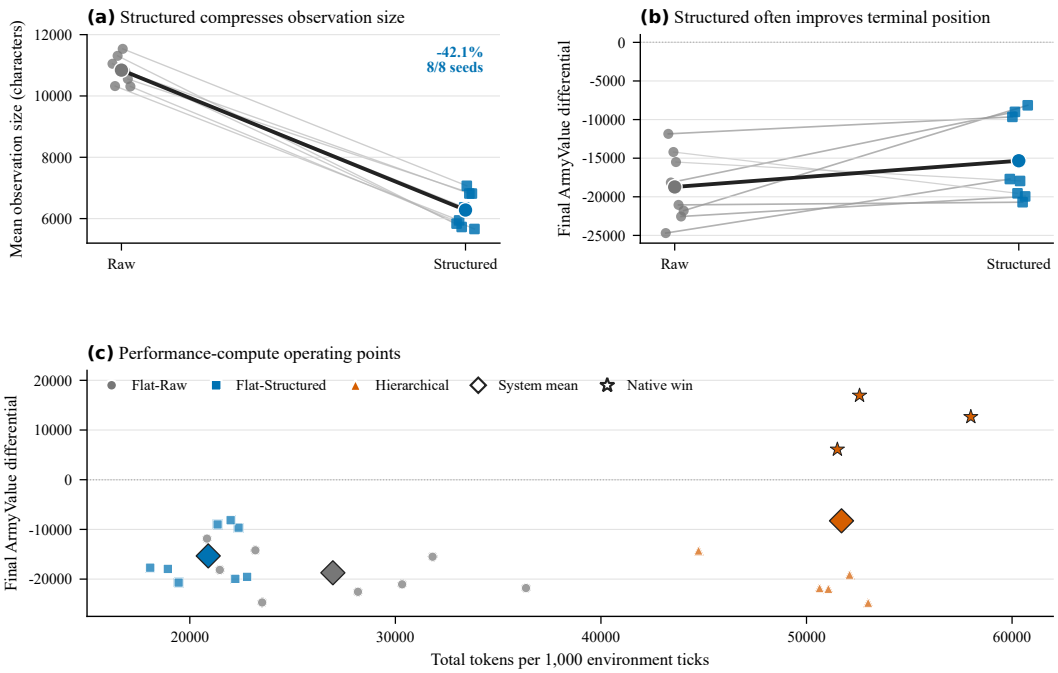


Figure 3: System-design effects in Full-Skirmish games. (a) Structured observation substantially reduces serialized input size and input tokens. (b) Under the same Flat architecture, Structured improves terminal ΔV in most paired games despite identical W/D/L outcomes. (c) Hierarchical combines lower-frequency high-level reasoning with higher-frequency lower-level control and achieves stronger long-term control while using substantially more tokens per 1,000 environment ticks.

We next compare Flat-Structured with Hierarchical to examine the changes associated with control architecture. Hierarchical organizes long-horizon strategy, production management, and local combat across different temporal scales, assigning different types of decisions to the corresponding control levels. In Figure 3(c), Hierarchical uses approximately 2.47× as many tokens per 1,000 environment ticks as Flat-Structured. Over the same comparison, mean ΔV rises from -15.3k to -8.3k, with wins increasing from 0/8 to 3/8.

The three systems therefore exhibit two distinct types of change. Moving from Raw to Structured primarily changes the observation representation, reducing input cost while improving terminal ΔV in most paired games. Moving from Structured to Hierarchical changes how computation is organized across decision types and temporal scales, increases overall model computation, and corresponds to stronger long-horizon control outcomes. Computational cost, information representation, and control architecture together define the system-design trade-off in Full-Skirmish.

3.4 Evaluation Breadth

The preceding experiments fix the primary model and opponent conditions to study temporal configuration and system design. Table 2 extends Full-Skirmish evaluation across model backbones, Bot behaviors, and opponent difficulties; within each experiment, only the variable under study is changed.

Table 2: Benchmark results under different model backbones, Bot behaviors, and opponent difficulties. Each condition uses 4 matched seeds. ΔV denotes the terminal difference in remaining ArmyValue between the two sides; the table reports the median, with brackets indicating the middle half of seed outcomes, in k units.

(a) Model backbone

Model	W/D/L	ΔV
GLM-5.3	2/0/2	-4.6 [-22.5, 13.7]
DeepSeek-V4.1-Flash	1/1/2	$+15.8$ [-16.2, 46.4]
Qwen3.8-Max	1/1/2	-2.8 [-15.8, 10.7]
Kimi-K3	0/0/4	-21.4 [-24.6, -17.5]

(b) Bot behavior

Bot	W/D/L	ΔV
Normal	2/0/2	-4.6 [-22.5, 13.7]
Rush	1/0/3	-4.8 [-6.7, 3.6]
Turtle	2/2/0	$+15.9$ [-9.4, 27.8]

(c) Difficulty

Setting	W/D/L	ΔV
Easy	4/0/0	$+17.8$ [15.6, 22.8]
Default	2/0/2	-4.6 [-22.5, 13.7]
Hard	0/0/4	-17.6 [-19.3, -17.2]

Different backbones already produce clearly different long-horizon control outcomes. DeepSeek and Qwen both obtain 1W/1D/2L, yet their median ΔV values are $+15.8$k and -2.8k, respectively. Identical discrete match outcomes can therefore correspond to substantially different continuous terminal states. This pattern also echoes the Flat-Raw / Flat-Structured comparison above: W/D/L captures the complete-task outcome, whereas ΔV preserves differences in game progress that are not visible from the final win/loss result alone.

Bot behavior changes the strategic tempo faced by the Agent. Rush, Normal, and Turtle are native OpenRA Bot presets emphasizing early aggression, standard play, and defensive development, respectively. The three behaviors produce different W/D/L and ΔV outcomes: Hierarchical obtains stronger long-horizon results against Turtle and weaker results against Rush. Its long-horizon strategy operates at a lower-frequency upper level, with local combat handled by higher-frequency lower levels. Turtle leaves a longer development phase, whereas Rush introduces local combat pressure earlier. Bot behavior therefore affects both the opponent's strategic pattern and the point at which the Agent must shift from long-horizon planning to high-frequency local response.

The difficulty sweep instead changes the overall strength of the same Normal Bot. Easy, Default, and Hard produce 4W/0D/0L, 2W/0D/2L, and 0W/0D/4L, with median ΔV decreasing from $+17.8$k to -4.6k and -17.6k. Whereas Bot behavior changes strategic tempo, difficulty shows how opponent strength under the same behavior corresponds to long-horizon control outcomes. Model backbone, Bot behavior, and difficulty thus provide three complementary evaluation axes within the same Full-Skirmish protocol, varying model reasoning capability, opponent behavior pattern, and opponent strength, respectively.

4 Related Work

4.1 Game and RTS Benchmarks

Games have long served as controlled environments for studying sequential decision-making, exploration, planning, and generalization, gradually giving rise to evaluation frameworks for long-horizon interaction and general Agent capabilities. (Cobbe et al., 2020; Küttler et al., 2020; Fan et al., 2022; Paglieri et al., 2025; Liu et al., 2024; Ma et al., 2024) These benchmarks extend evaluation beyond single-step prediction to sustained sequences of observation, decision, and action, using multi-round interaction and process-level metrics to characterize behavioral progress beyond final outcomes. (Liu et al., 2024; Ma et al., 2024) With the development of LLM and VLM Agents, systems such as MineDojo, Voyager, BALROG,

Cradle, SIMA, and VideoGameBench further use games to study open-ended tasks, skill acquisition, long-horizon planning, and cross-environment control. (Fan et al., 2022; Wang et al., 2024; Paglieri et al., 2025; Tan et al., 2025; SIMA Team et al., 2024; Zhang et al., 2025) Our work follows this process-level evaluation perspective, focusing on the temporal process connecting observation, model inference, and action application in continuously running environments.

Real-time strategy games add continuously changing environments, resource management, unit control, and long-horizon decision-making to this setting. StarCraft II has established a mature foundation for research on complex real-time decisions, spanning standardized environments, reinforcement learning, and complete games. (Vinyals et al., 2017; 2019) More recent work introduces LLMs into RTS settings to study strategic decision-making with textual observations, relatively complete action spaces, and full games. (Ma et al., 2024; Li et al., 2025; Shen et al., 2025) These studies primarily characterize economic management, production, combat, and long-horizon planning through behavior observed in complete tasks and their final outcomes.

We adopt RTS as a substrate for sustained strategic decision-making while bringing the real-time interaction process itself into evaluation. OpenRA-RL already provides a real-time Agent interface built on the OpenRA game substrate. (OpenRA-RL Contributors, 2026) Within the same substrate, Fixed-Force offers controlled real-time combat diagnostics, whereas Full-Skirmish retains economy, production, technology, and long-horizon combat, linking short-horizon temporal diagnostics to complete strategic control. Existing RTS benchmarks more commonly treat complete tasks or games as the primary evaluation unit; research on real-time Agents instead examines decision-making in continuously evolving environments through response time, time budgets, and asynchronous interaction. Bringing these questions into a shared RTS substrate makes it possible to observe local temporal effects under controlled conditions while evaluating complete long-horizon strategic control within the same environment. Evaluation can thus extend beyond isolated responses or terminal outcomes to the ongoing control process connecting individual responses to long-horizon outcomes.

4.2 Real-Time Agents in Evolving Environments

Research on Agents in dynamic environments has increasingly treated time as part of the control process. When the environment continues to advance during model inference, each decision spans wall-clock time, environment evolution, and subsequent decision opportunities. Prior work has examined the relationships among inference time, response speed, time budgets, and decision quality across continuously changing embodied environments, real-time reasoning, high-frequency decision-making, real-time games, and cross-environment control. Related studies further consider asynchronous interaction and decision organization across multiple temporal scales. (Zhou et al., 2024; Wen et al., 2026; Kang et al., 2025; Zhang et al., 2025; SIMA Team et al., 2024; Tang et al., 2026; Shraga et al., 2026)

Different approaches handle the relationship between model inference and environment progression in different ways. VideoGameBench pauses the game during action generation, evaluating model decisions under a frozen environment state. (Zhang et al., 2025) Continuously running settings preserve the coupling among wall-clock response time, environment progression, and decision opportunity. Our benchmark jointly records wall-clock response time, observation-to-action gap, decision frequency, and final control outcomes in a real-time RTS, while added waiting and world speed vary the mapping between wall-clock time and environment time. Fixed-Force analyzes these temporal quantities in controlled combat states. Full-Skirmish places the same observation–Agent–action loop back into a complete long-horizon strategic process, allowing temporal configuration to be evaluated alongside observation representation, control architecture, and model backbone as components of the complete Agent system.

5 Discussion

Our results point to a broader problem in real-time control: how an Agent should allocate limited time, information, and computation in a continuously evolving environment. Model inference takes place inside the control loop, so computational cost is reflected not only in wall-clock time but also in the environment state reached by the time an action is applied, with further consequences for subsequent observations and decision opportunities. Observation representation determines how much information must be processed for each decision; control architecture determines how computation is distributed across tasks and temporal scales; environment dynamics determine how far the world advances while that computation is taking place. Time, information, and computation are therefore not independent design variables, but interacting components of the same closed-loop control process. Structured observation organizes information; Hierarchical control organizes computation. Together, they illustrate how real-time Agent performance depends on converting limited computation into sustained control over a changing state.

This closed-loop perspective also shapes how real-time Agents should be evaluated. Wall-clock response time, observation-to-action gap, and decision frequency capture different temporal aspects of the control process. Continuous state measures such as ΔV reflect outcomes accumulated over multiple rounds of interaction, whereas W/D/L captures the final result at the level of the complete task. Fixed-Force and Full-Skirmish further provide two complementary evaluation scales: the former isolates local temporal effects, while the latter evaluates complete long-horizon control across economy, production, scheduling, and combat. Together, the benchmark tasks and metrics form an analytical chain from local interaction processes to long-horizon strategic outcomes, allowing system differences to be located at specific stages of the control trajectory rather than appearing only as a single terminal score.

Under this framework, the object of evaluation is the complete Agent–environment closed loop. The model backbone provides reasoning capability; the observation determines how information enters the model; the architecture organizes computation; temporal configuration governs the relationship between that computation and environment progression; opponent conditions define the external control pressure under which the system operates. Different designs may therefore reach similar terminal outcomes through different interaction trajectories, or produce substantially different long-horizon control behavior under similar computational budgets. These comparisons ultimately reduce to the same question: how an Agent system organizes limited time, information, and computation into a stable and effective process of sustained decision-making. By placing environment progression, information representation, computation organization, interaction processes, and long-horizon outcomes within a common experimental framework, the benchmark extends comparison of real-time Agents beyond isolated model capability or final match outcome toward systematic analysis of the complete control process.

6 Conclusion

We introduce an OpenRA-based benchmark for studying LLM Agents in continuously running real-time strategic environments. Under a unified protocol, the benchmark records wall-clock response time, observation-to-action gap, decision frequency, and control outcomes, using Fixed-Force diagnostics and Full-Skirmish games to examine controlled temporal effects and long-horizon strategic control, respectively. Experiments show that added waiting changes both the observation-to-action gap and decision frequency, while the same wall-clock waiting time can correspond to substantially different gaps under different world speeds. Structured observation reduces input cost, whereas Hierarchical control achieves stronger long-horizon performance with greater model computation. The benchmark provides a unified experimental basis for temporal evaluation and system-level comparison of real-time strategic Agents.

AI use statement

Generative AI tools were used for code implementation and debugging, discussion of experimental design, translation, literature search and organization, analysis of experimental results, and organization, revision, and language polishing of the paper draft. All experimental configurations, results, citations, technical descriptions, and conclusions were verified by the authors, who take full responsibility for the final content of the paper.

Reproducibility statement

The benchmark uses containerized OpenRA, fixed experimental configurations, and explicit random seeds. The Agent and prompt configurations, environment and task settings, complete experimental conditions, and per-seed results required for reproduction are provided in the Appendix submitted together with the supplementary materials. The supplementary package also includes the benchmark source code, experiment configurations, result data, and reproduction instructions.

References

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents. In International Conference on Learning Representations, 2024.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In Proceedings of the 37th International Conference on Machine Learning, 2020.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The NetHack Learning Environment. In Advances in Neural Information Processing Systems, 2020.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge. In Advances in Neural Information Processing Systems, Datasets and Benchmarks Track, 2022.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. BALROG: Benchmarking Agentic LLM and VLM Reasoning on Games. In International Conference on Learning Representations, 2025.

Weiyu Ma, Qirui Mi, Yongcheng Zeng, Xue Yan, Yuqiao Wu, Runji Lin, Haifeng Zhang, and Jun Wang. Large Language Models Play StarCraft II: Benchmarks and a Chain of Summarization Approach. In Advances in Neural Information Processing Systems, 2024.

Zongyuan Li, Yanan Ni, Runnan Qi, et al. LLM-PySC2: StarCraft II Learning Environment for Large Language Models. In Advances in Neural Information Processing Systems, 2025.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, et al. StarCraft II: A New Challenge for Reinforcement Learning. arXiv:1708.04782, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, et al. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. Nature, 575:350–354, 2019.

Pengbo Shen, Yaqing Wang, Ni Mu, et al. SC2Arena and StarEvolve: Benchmark and Self-Improvement Framework for LLMs in Complex Decision-Making Tasks. arXiv:2508.10428, 2025.

Qinhong Zhou, Sunli Chen, Yisong Wang, et al. HAZARD Challenge: Embodied Decision Making in Dynamically Changing Environments. In International Conference on Learning Representations, 2024.

Yule Wen, Yixin Ye, Yanzhe Zhang, Diyi Yang, and Hao Zhu. Real-Time Reasoning Agents in Evolving Environments. In International Conference on Learning Representations, 2026.

Hao Kang, Qingru Zhang, Han Cai, Weiyuan Xu, Tushar Krishna, Yilun Du, and Tsachy Weissman. Win Fast or Lose Slow: Balancing Speed and Accuracy in Latency-Sensitive Decisions of LLMs. In Advances in Neural Information Processing Systems, 2025.

Alex L. Zhang, Thomas L. Griffiths, Karthik R. Narasimhan, and Ofir Press. VideoGameBench: Can Vision-Language Models Complete Popular Video Games? arXiv:2505.18134, 2025.

Zineng Tang, Kelsey R. Allen, Sjoerd van Steenkiste, Ishita Dasgupta, and Alane Suhr. Decoupling Planning and Control for Instructable Agents. In Conference on Language Modeling, 2026.

Ilai Shraga, Roei Eshel, and Lior Gorelik. Approved Too Late: Verdict Staleness in LLM-Guarded Self-Adaptive Systems. arXiv:2608.26306, 2026.

OpenRA-RL Contributors. OpenRA-RL: Open Framework for AI Agents to Play Red Alert through Reinforcement Learning. GitHub repository, 2026.

Chang Ma, Junlei Zhang, Zhihao Zhu, et al. AgentBoard: An Analytical Evaluation Board of Multi-turn LLM Agents. In Advances in Neural Information Processing Systems, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, et al. Voyager: An Open-Ended Embodied Agent with Large Language Models. Transactions on Machine Learning Research, 2024.

Weihao Tan, Wentao Zhang, Xinrun Xu, et al. Cradle: Empowering Foundation Agents Towards General Computer Control. In International Conference on Machine Learning, 2025.

SIMA Team et al. Scaling Instructable Agents Across Many Simulated Worlds. arXiv:2404.10179, 2024.