HELIOS - Holistic Experiment Learning Intelligent Orchestration System

Sissi Feng [1] **Yuchen Han** [1] **Yang Bai** [1]

[1] *Acceleration Consortium, University of Toronto, 700 University Ave., Toronto, ON M7A 2S4, Canada*. Correspondence to: Sissi Feng sissi.feng@utoronto.ca.

1. Introduction

Self-driving laboratories close the loop between hypothesis, robotic execution, and model updating, yet most platforms orchestrate this loop as a monolithic script: one optimiser picks the next experiment, one compiler emits machine code, and execution proceeds sequentially [1]. When any stage fails—a contaminated well, an optimiser stuck in a local basin, a mislabelled spectrum—the entire pipeline stalls. The architecture cannot reflect on its own performance, nor can it adapt the way it searches as evidence accumulates.

HELIOS is designed to address each of these limitations explicitly. To prevent cascade failures, we decompose the pipeline into isolated layers, each owning a narrow decision scope and communicating through typed message contracts. Thus, a contaminated well triggers localised recovery rather than a full abort. To handle the diverse expertise required across a discovery campaign, we introduce ephemeral specialist agent swarms (Scientist, Engineer, Analyst, Validator) that are spawned on demand and disbanded after their task, bringing the right reasoning to each stage without bottleneck. And to enable the system to improve not only what it proposes but how it searches, we embed a dual-loop reinforcement learning architecture: an inner loop that selects candidate-generation strategies per experimental round, and an outer meta-loop that refines the orchestrator's own scheduling policy across campaigns. Together, these three principles are realised in a four-layer architecture we call HELIOS (Fig. 1).

2. Framework

Layer 3 Orchestrator. A central Orchestrator accepts a declarative *TaskContract* (objectives, search space, safety envelope, stopping conditions) and coordinates round-by-round campaign execution. Unlike a static scheduler, the orchestrator dynamically spawns four specialist agent swarms when their expertise is needed: a *Scientist* swarm for hypothesis generation via first-principles reasoning, an *Engineer* swarm for protocol optimisation and in-silico simulation, an *Analyst* swarm for real-time spectral and image interpretation, and a *Validator* swarm for hallucination detection—verifying that agent-proposed parameters and interpreted results are physically plausible before they enter the decision loop. Swarms are ephemeral: spawned on demand, disbanded after their task, and re-spawnable with updated context in the next round.

Layers 2–0 Planning, Compilation, Execution. The Planning layer (L2) maps the contract onto a multi-round schedule and generates candidate parameter sets through pluggable strategies (Latin hypercube sampling, Bayesian optimisation with EI/UCB acquisition, prior-guided sampling). The Compilation layer (L1) converts abstract candidates into executable Opentrons OT-2 protocols via deck-layout optimisation and syntax-safe code generation, with a natural-language interface. The Execution layer (L0) manages parallel robot control through a Hardware Abstraction Layer (HAL) that exposes a unified bidirectional API, decoupling all upper layers from specific instrument drivers—liquid handling, potentiostat, camera, or temperature modules can be swapped without modifying agent logic.

Cross-cutting concerns. A Safety Agent holds veto authority at every layer transition, enforcing preflight resource checks and runtime constraint monitoring. A Recovery Agent implements exponential-backoff retry with fault-injection testing. All communication flows through a publish–subscribe event bus that drives three asynchronous feedback listeners: a *Memory* listener backed by vector search, a knowledge graph, and an immutable experiment ledger for causal retrieval over past campaigns; a *Reviewer* listener that scores run quality; and an *Evolution* listener that tightens prior distributions and archives high-scoring protocol templates.

Dual-loop reinforcement learning. The central design thesis of HELIOS is that a self-driving lab should improve not only the experiments it proposes but also the *process by which it proposes them*. We implement this through a two-tier RL architecture.

The inner loop operates per round. A Q-learning agent with adaptive state discretisation (18→173 states as data accumulates) selects among five candidate-generation strategies; a DQN variant with target networks provides a neural alternative. Shaped rewards combine improvement magnitude, convergence progress, and exploration–exploitation balance.

The outer loop operates per campaign. A Metrics Engine tracks three KPIs—*discovery velocity* (objective improvement per unit time), *cost-per-insight* (experiments per actionable finding), and *success rate*—and feeds them to the orchestrator as a meta-optimisation signal. The orchestrator responds by adjusting round cadence, reallocating budget across swarm groups, or escalating to human-gate approval when confidence drops. Across successive campaigns on related tasks, the outer loop accumulates a policy that front-loads effective strategies and avoids previously unproduc-

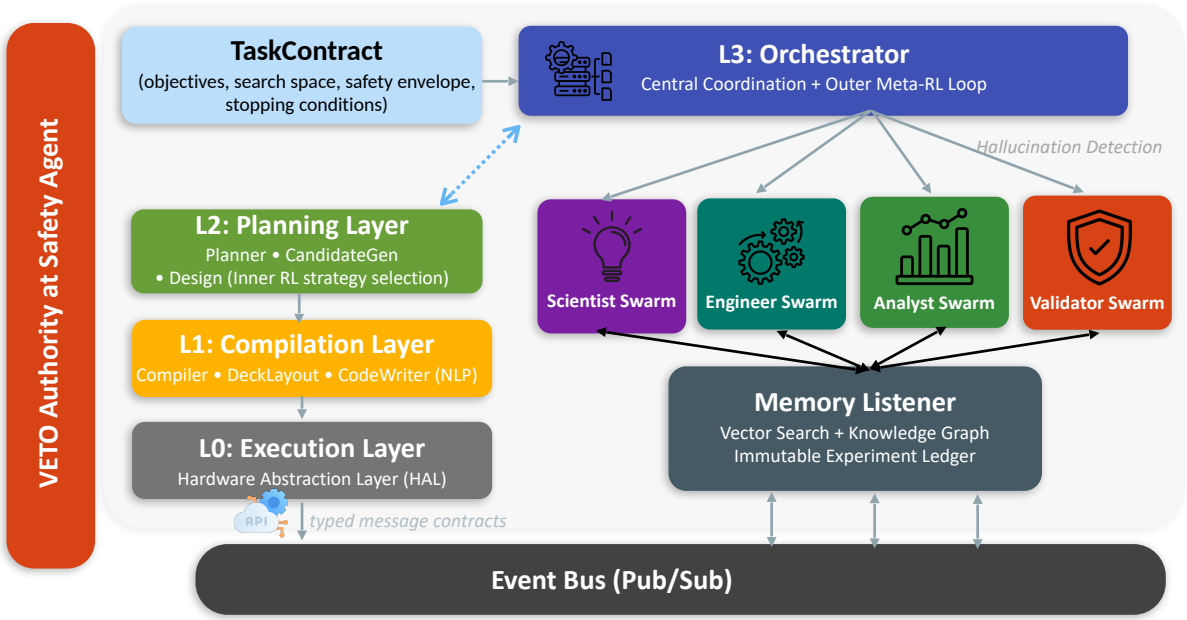

Fig. 1: HELIOS Architecture Overview. *Left*: TaskContract feeds into four asynchronous layers (L2: Planning, L1: Compilation, L0: Execution) connected by typed message contracts. A cross-cutting Safety Agent (red vertical bar) holds veto authority at every transition; the Hardware Abstraction Layer in L0 decouples upper logic from instrument drivers. *Centre*: The L3 Orchestrator centrally coordinates and spawns four specialist swarms on demand (Scientist, Engineer, Analyst, Validator with hallucination detection), routing them via a publish–subscribe event bus. *Bottom*: Persistent memory (vector search + knowledge graph + immutable experiment ledger) enables cross-campaign learning, while dual-loop RL runs inside the architecture—an inner loop in L2 selects candidate-generation strategies per round and an outer meta-RL loop in L3 optimises the orchestrator's own scheduling policy through discovery-velocity, cost-per-insight, and success-rate feedback.

tive regions of strategy space.

Validation. We validate HELIOS on a hydrogen-evolution-reaction (HER) catalyst discovery campaign targeting overpotential $\eta_{10} < 50\,\text{mV}$ across a four-component precursor space. Over 24 automated rounds, the system transitions from space-filling exploration through Bayesian exploitation to prior-guided refinement, with the dual-loop RL matching or outperforming fixed-strategy baselines. Ablation experiments isolate the contributions of the outer meta-loop, agent-swarm spawning, and three-tier convergence detection (majority-voting, oscillation/SNR analysis, Bayesian change-point). Contract versioning with checksum verification provides full provenance.

2.1 Related work

Autonomous platforms have evolved from scripted workflows [2] through human-gated modular systems [3] to LLM-driven agents such as Coscientist [4] and hierarchical multi-agent designs like ChemAgents [5]. HELIOS differs in three respects: (i) *adaptive orchestration*—the outer RL loop lets the system improve its own decision-making process, whereas existing platforms use fixed scheduling; (ii) *ephemeral expert swarms with hallucination detection*—no prior SDL incorporates a Validator swarm that verifies agent outputs against physical plausibility before execution; and (iii) *hardware-agnostic design*—the HAL ensures that adding a new instrument requires only a driver, not changes to planning or safety logic.

Acknowledgments

This research is part of the University of Toronto's Acceleration Consortium, which receives funding from the Canada First Research Excellence Fund (CFREF-2022-00042).

References

[1] Milad Abolhasani and Eugenia Kumacheva. The rise of self-driving labs in chemical and materials sciences. *Nature Synthesis*, 2:483–492, 2023.

[2] Sebastian Steiner, Jakob Wolf, Stefan Glatzel, Anna Andreou, Jarosław M. Granda, Graham Keenan, Trevor Hinkley, Gerardo Aragon-Camarasa, Philip J. Kitson, Davide Angelone, and Leroy Cronin. Organic synthesis in a modular robotic system driven by a chemical programming language. *Science*, 363(6423):eaav2211, 2019.

[3] B. P. MacLeod, F. G. L. Parlane, T. D. Morrissey,

F. Häse, L. M. Roch, K. E. Dettelbach, R. Moreira, L. P. E. Yunker, M. B. Rooney, J. R. Deeth, V. Lai, G. J. Ng, H. Situ, R. H. Zhang, M. S. Elliott, T. H. Haley, D. J. Dvorak, A. Aspuru-Guzik, J. E. Hein, and C. P. Berlinguette. Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances*, 6(20):eaaz8867, 2020.

[4] Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

[5] Tao Song, Man Luo, Linjiang Chen, Yan Huang, Qing Zhu, Daobin Liu, Baicheng Zhang, Gang Zou, Fei Zhang, Weiwei Shang, Jun Jiang, and Yi Luo. A multiagent-driven robotic AI chemist enabling autonomous chemical research on demand. *Journal of the American Chemical Society*, 147(15):12534–12545, 2025.