Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal

Cite this: DOI: 10.1039/d6dd00004e

PRISM: protocol refinement through intelligent simulation modeling

Brian Hsu, †[ab] Priyanka V. Setty, †[ab] Rory M. Butler, †[ab] Ryan Lewis, [a] Casey Stone, [a] Rebecca Weinberg, [a] Thomas Brettin, [ab] Rick Stevens, [ab] Ian Foster [ab] and Arvind Ramanathan *[a]

Automating experimental protocol design and execution remains as a fundamental bottleneck in realizing self-driving laboratories. We introduce PRISM (Protocol Refinement through Intelligent Simulation Modeling), a framework that automates the design, validation, and execution of experimental protocols on a laboratory platform composed of off-the-shelf robotic instruments. PRISM uses a set of language-model-based agents that work together to generate and refine experimental steps. The process begins with automatically gathering relevant procedures from web-based sources describing experimental workflows. These are converted into structured experimental steps (*e.g.*, liquid handling steps, deck layout and other related operations) through a planning, critique, and validation loop. The finalized steps are translated into the Argonne MADSci protocol format, which provides a unified interface for coordinating multiple robotic instruments (Opentrons OT-2 liquid handler, PF400 arm, Azenta plate sealer and peeler) without requiring human intervention between steps. To evaluate protocol-generation performance, we benchmarked both single reasoning models and multi-agent workflow across constrained and open-ended prompting paradigms. The resulting protocols were validated in a digital-twin environment built in NVIDIA Omniverse to detect physical or sequencing errors before execution. Using Luna qPCR amplification and Cell Painting as case studies, we demonstrate PRISM as a practical end-to-end workflow that bridges language-based protocol generation, simulation-based validation, and automated robotic execution.

Received 6th January 2026
Accepted 28th April 2026

DOI: 10.1039/d6dd00004e

rsc.li/digitaldiscovery

1 Introduction

The automation of experimental protocol design and execution remains a fundamental bottleneck in realizing self-driving laboratories. While robotic platforms are increasingly available, translating scientific intent into executable laboratory protocols requires substantial domain expertise, as the process is error-prone and demands intimate knowledge of both experimental procedures and instrument-specific formatting requirements. Existing protocol description languages provide structured representations but remain static and hardware-specific, requiring manual rewriting when laboratory configurations change. Recent work has shown that large language models (LLMs) can generate plausible experimental procedures from natural language descriptions, yet these outputs often contain parameter underspecification, physical infeasibility, or sequencing errors that would cause failures during execution. Meanwhile, simulation and digital twin technologies have advanced laboratory monitoring and documentation but have not been integrated into the protocol generation pipeline as effective pre-execution validation tools. Direct testing of unvalidated protocols on physical hardware is costly, time-consuming, and risks both equipment damage and material waste. These challenges motivate the development of systems that combine automated protocol generation with rigorous pre-execution validation to enable safe, reliable, and fully autonomous laboratory operation.

1.1 Related work in protocol automation

Early efforts to formalize laboratory procedures focused on creating machine-readable descriptions of experimental steps. XDL[1] demonstrated this for automated chemistry by expressing operations such as heating or adding reagents in a hardware-agnostic language. Autoprotocol[2] extended this idea to cloud-operated labs more broadly, introducing a JSON-based specification for generic laboratory actions like pipetting, incubating, sealing, and measuring so that protocols could be executed across diverse automation systems. More integrated platforms such as aquarium[3] coupled protocol specification with inventory tracking and execution oversight, using a high-level

[a]Argonne National Laboratory, 9700 S. Cass Avenue, Lemont, IL 60439, USA. E-mail: ramanathana@anl.gov
[b]Department of Computer Science, University of Chicago, Chicago, IL 60637, USA
† These authors contributed equally.

language and Laboratory Information Management System (LIMS) to support technician-in-the-loop workflows.

While these systems illustrate a clear progression toward more structured and reproducible experimentation, they are all limited in important respects. In particular, protocol descriptions are static, tied to specific configurations, and offer little support for validating physical feasibility before execution. They also require specialized technical expertise to develop and adapt. These gaps motivate the need for more adaptive approaches that combine structured representations with contextual reasoning and automated validation.

LLMs have shown the ability to generate multi-step experimental workflows when used alongside external tools. Recent surveys have cataloged over 260 such "scientific LLMs" across disciplines,[4] classifying them as "autonomous agents" that combine reasoning, planning, and tool use to solve complex scientific tasks.[5,6] ChemCrow[7] demonstrated LLM integration in chemistry workflows by coupling an LLM with reaction databases and planners, allowing high-level goals to be decomposed into executable steps. Coscientist[8] advanced further by integrating documentation search to autonomously execute reactions.

More recently, hierarchical multi-agent systems like ChemAgents[9] and LLM-RDF[10] have demonstrated end-to-end chemical discovery, though they notably rely on human-in-the-loop mechanisms or manual verification to ensure safety prior to execution. Building on this idea, BioPlanner[11] examined whether similar capabilities extend to biological protocols, evaluating the completeness and clarity of model-generated procedures and conducting limited feasibility tests. Similarly, Inagaki *et al.*[12] and Alchemist[13] utilized LLMs to generate executable robotic scripts (*e.g.*, for Opentrons), demonstrating that while syntax errors can be caught *via* standard API simulators, physical execution failures persist without robust environmental simulation.

Yet more recent work, such as ProtoCode,[14] moves toward making outputs actionable by introducing structured intermediate representations that bridge free-text descriptions and machine-interpretable robotic actions. Agentic Lab[15] employs multi-agent orchestration for knowledge retrieval, protocol composition, and iterative refinement through LLM-based self-reflection and human feedback.

Despite this progress, key limitations persist. Parameter underspecification is common: model outputs often omit key details such as reagent volumes, concentrations, incubation times, or mixing cycles. Some instructions are physically infeasible, for instance, suggesting movements of labware that exceed the reach of a robotic arm or stacking plates in ways incompatible with the deck layout. Recent benchmarking on laboratory safety has revealed that even state-of-the-art models fail to identify critical hazards in realistic scenarios over 30% of the time.[16] Most outputs remain in natural language or semi-structured pseudocode, requiring manual translation into robot-specific formats. Finally, current evaluations rely mainly on expert review rather than simulation or digital-twin testing; Agentic Lab provides real-time feedback during physical execution through AR glasses and vision-language models, but validates procedures during execution rather than simulating potential outcomes beforehand. PRISM addresses these gaps through multi-agent planning with iterative critique to resolve underspecification and sequencing errors (stage 1, Section 2.1) and reasoning-model-driven translation into robot-executable formats (stage 2, Section 2.2).

Together, these findings suggest that while LLMs can assist in translating high-level scientific intent into procedural steps, their outputs still need systematic refinement and physical validation. Bridging this gap requires combining language generation with simulation, contextual reasoning, and platform-aware validation—an approach that motivates the PRISM framework developed in this work.

1.2 Simulations and digital twins

Digital twins (in this context: virtual replicas of physical laboratory systems) provide opportunities to design, test, deploy, monitor, and control real-world robotic processes.[17] Such simulation environments enable offline testing and validation of robotic operations through physics-based collision detection, joint constraints, and realistic motion planning. However, most applications of digital twins for laboratory automation only emphasize post-hoc documentation, execution tracing, and real-time monitoring, rather than pre-execution validation and refinement. Recent reviews in catalysis have highlighted this gap, noting that while LLMs offer flexibility, the development of digital twins to ensure the physical safety and integrity of generated protocols remains in its infancy.[18]

Previous advances in LLM applications have demonstrated certain key but limited pre-execution validation and refinement techniques. For example, CLAIRIFY[19] developed an approach using verifier-assisted iterative prompting to generate valid programs in domain-specific languages. By combining automatic iterative prompting with syntax and rule-based program verification, CLAIRIFY ensures valid programs that incorporate environmental constraints. However, while this verification checks syntactic correctness and certain manually defined rules, it does not comprehensively validate physical feasibility, collision risks, or hardware constraints through high-fidelity simulation. ORGANA,[20] building on CLAIRIFY, executes protocols directly on physical hardware, solving task and motion planning problems on the fly as part of the protocol's execution process. Similarly, Agentic Lab's real-time monitoring through vision-language models enables continuous learning and protocol refinement based on post-execution analysis, but it does not provide the grounded pre-execution validation that simulation-based approaches offer. These system's abilities to detect and correct errors depends on syntax verification, handcrafted rule-based verification, and real-time visual observation during physical execution, rather than anticipating and preventing errors through prior simulation. PRISM addresses this gap by positioning physics-based simulation as a mandatory pre-execution validation gate within the protocol generation loop, creating a closed feedback loop where detected errors inform iterative refinement (stage 2, Section 2.3).

1.3 Contributions and goals

A critical gap exists in the current landscape of laboratory automation: no existing work combines end-to-end LLM-driven protocol generation and simulation-based error detection into a unified pipeline for fully autonomous generation of trustworthy protocols. Digital twin work provides comprehensive monitoring and tracing capabilities but remains insufficiently integrated with automated protocol generation systems.

The PRISM framework addresses this gap through a three-stage pipeline that integrates multi-agent LLM planning, reasoning-model-driven protocol generation, and physics-based simulation error detection. PRISM positions simulation as a mandatory pre-execution gate, creating a closed feedback loop where detected errors inform iterative protocol refinement. We demonstrate the system's effectiveness through PCR amplification (experimentally validated on our robotic platform) and Cell Painting (computationally validated), while providing comparative analysis across multiple state-of-the-art language models. This work represents a step toward fully autonomous experimental design and execution, bridging the divide between AI-generated plans and safe, reliable laboratory automation.

2 Methods

We introduce PRISM: Protocol Refinement through Intelligent Simulation Modeling, a framework that automates the design, validation, and execution of experimental protocols on a laboratory platform composed of off-the-shelf robotic instruments. We use a multi-agent framework (Fig. 1, stage 1), described in Section 2.1, to generate the steps of an experimental procedure of interest, initially described in plain English. The steps follow a structured format that the protocol generator recognizes and can convert into robot-compatible instructions. The protocol generator (Fig. 1, stage 2) described in Section 2.2 combines these structured steps with feedback from a digital twin simulation described in Section 2.3, iteratively refining the design until a valid experimental protocol is produced. The final valid protocol is then ready to execute in the real-world laboratory (Fig. 1, stage 3) as described in Section 2.4.

2.1 Protocol planning

The protocol planning stage focuses on generating complete structured liquid handling instructions in plain English that capture all the reagents, labware, and procedural details required to execute an experiment on an automated platform. PRISM targets experimental workflows that are performed repeatedly in laboratory settings, such as PCR, Cell Painting, and other standard assays, where robotic automation offers the greatest benefit in terms of consistency, throughput, and reduction of human error. The input to the planning stage can be retrieved automatically *via* the WebSurfer agent from online sources, or provided directly by the researcher from an existing bench protocol, making the framework applicable even when published procedures are limited or unavailable. These steps

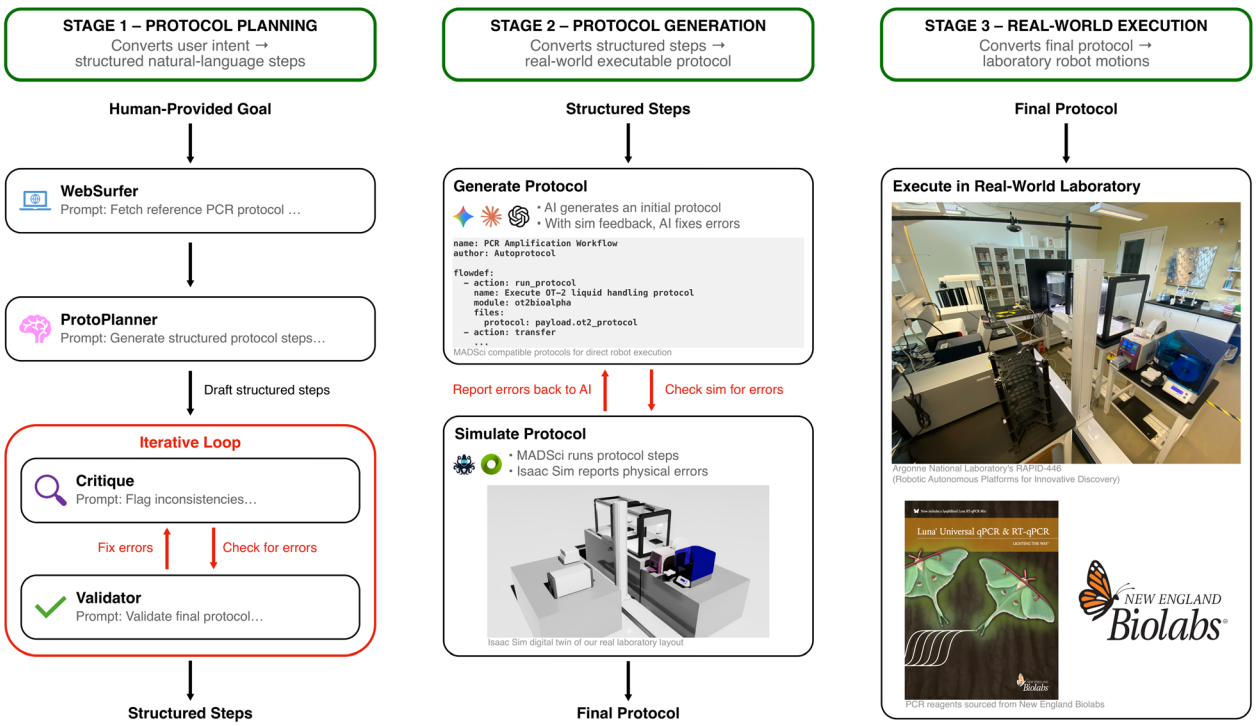

Fig. 1 Overview of the PRISM framework for protocol generation and execution. The system consists of three main stages: protocol planning, where user intent is converted into structured steps; protocol generation, where structured English instructions are transformed into robot-aware actions and iteratively refined through validation cycles in Omniverse before execution; and real-world execution, where the full pipeline is validated using the Luna qPCR protocol in our autonomous laboratory.

are not directly robot-executable, but are formatted to preserve the logical sequence of actions and physical constraints necessary for robotic translation.

We evaluate two complementary prompting paradigms, constrained and open-ended, for which the full prompt templates are provided in the SI. In the constrained setting, models receive explicit reagent volumes, well layouts, and step-format rules, whereas open-ended prompting provides only high-level experimental goals, requiring models to infer volumes, mappings, and transfer sequences. We test each prompting paradigm across two architectural frameworks—a multi-agent system and a single-agent system to assess each model's ability to generate, refine, and self-correct experimental protocols over three iterative refinement cycles.

2.1.1 Multi- *vs.* single-agent protocol planning frameworks. We define an agent as a single LLM invocation with a dedicated system prompt that constrains its role to a specific subtask (*e.g.*, retrieval, planning, critique, or validation). Agents communicate sequentially: the output of one agent is provided as input to the next. All agents use the same underlying model but operate under different instructions and context. In the single-agent framework, one model invocation with a single combined prompt performs all roles within a continuous chain of thought. We compare two architectural approaches for protocol planning: multi-agent and single-agent. These two frameworks differ primarily in how reasoning is distributed and how errors are detected and corrected.

In the multi-agent system, the task is decomposed across four specialized agents, each with a distinct role. The WebSurfer agent retrieves experimental information from online sources and extracts procedural steps and reagents. The protocol planner agent then converts this unstructured description into a structured plan, selecting labware, assigning reagents to specific deck positions, and rewriting each action into a consistent liquid-handling schema. Next, a critique agent evaluates the structured steps for missing information, volume inconsistencies, formatting issues, and logical breakdowns. Finally, the validator agent applies any needed corrections and re-submits updated steps to the critique agent, establishing an iterative refinement loop. Because each agent reasons only within its specialized scope, the system benefits from modularity: reasoning is constrained, errors are easier to isolate, and corrective feedback is more targeted. However, this setup introduces dependencies between agents, and failures in early stages can propagate unless caught by downstream checks.

In contrast, the single-agent (reasoning model) framework relies on a single model to perform all stages of the task within one continuous chain of thought. The model must retrieve relevant experimental details, infer appropriate deck layouts, assign reagents, compute volumes, and generate a fully enumerated set of structured liquid-handling steps in a single pass, without external critique or role decomposition. This approach benefits from simplicity and faster execution but places a significantly higher cognitive load on the model: all planning, validation, error detection, and correction must occur internally. Without explicit modular boundaries, the model may overlook inconsistencies, lose track of earlier constraints, or produce logically correct but physically invalid steps, especially as workflows become longer and more complex.

Overall, the multi-agent system distributes reasoning across specialized components that reinforce each other, whereas the single-agent system must manage the entire reasoning space at once. As we explain in more detail below, we find that the single-agent system works reasonably well for short protocols with few dependencies, such as PCR, where the sequence is largely fixed (prepare reactions → thermocycler) and the exact order of reagent additions is not critical. In contrast, workflows like Cell Painting have many interdependent steps where timing, order, and reagent interactions matter, making them more prone to cascading reasoning errors when handled by a single model. For these complex, more structured protocols, the multi-agent framework is more robust and consistently produces correct outputs.

2.1.2 Evaluation setup. We benchmark the two frameworks under constrained and open-ended prompting using five state-of-the-art LLMs: GPT-5,[21] Claude Opus 4.1,[22] Claude Sonnet 4.5,[23] Gemini 2.5 Pro,[24] and Gemini 2.5 Flash.[24] We evaluated each model–framework combination under both constrained and open-ended prompting configurations for up to three refinement iterations. In cases where convergence was not achieved within three iterations, the final attempt was recorded for analysis. For the PCR experiment, the GPT-5 constrained multi-agent protocol was used as ground truth, as it produced a fully correct protocol without requiring any correction cycles and was validated by an experimental biologist. Each generated protocol was compared against this reference across predefined ground truth categories like logical transfers, master mix calculations, reagent and well assignments, pipette selection, and step formatting to identify correct, missing, and extraneous steps (true positives, false negatives, and false positives). F_1 scores were then computed within and across these categories to quantify generation accuracy.

2.2 Initial protocol generation

To translate from English protocols into YAML format, we used LLMs with reasoning capabilities that perform a thinking step before generating responses. We tested this stage of the framework with six reasoning models: GPT-5,[21] Claude Opus 4.1,[22] Claude Sonnet 4.5,[23] Gemini 2.5 Pro,[24] Gemini 2.5 Flash,[24] and Gemini 2.5 Flash-Lite.[25]

All model generation parameters were kept at their default values for consistency and simplicity. When provided with input from the protocol planning stage, these models convert high-level English instructions into structured YAML output by breaking down the instructions into specific robot commands that can be executed by laboratory automation systems. The resulting YAML files contain sequentially ordered robot actions along with their associated parameters, which together form a complete machine-readable protocol.

The complexity of laboratory workflows makes simple rule-based translations from English to robot commands insufficient for many scenarios. What appears as a straightforward instruction in English, such as "seal the plate," often requires

multiple coordinated robot actions, including retrieving the plate from its current location, transferring it to a sealing robot, and executing the sealing operation. This complexity increases when working with robots that have specific operational requirements, such as those that must be opened before plates can be inserted and closed before operations can be performed, or when unique position and rotation requirements of the source and destination robots conflict. When laboratories incorporate multiple robots with different operational constraints and spatial restrictions, the sequence of actions becomes non-trivial to determine, particularly when synchronization between robots is necessary. A single misplaced step in the sequence can lead to physical impossibilities, such as attempting to place a plate in a closed device or trying to operate a robot before proper preparation. These challenges make reasoning about the entire workflow important, as the model must understand not only each individual robot's capabilities but also how they interact across the laboratory's spatial and operational constraints.

2.2.1 Model input. The translation process uses three main input components to generate robot-executable protocols. First, the system receives English process steps with scientific parameters such as volumes, temperatures, and timing information derived from the previous protocol planning stage. These steps describe the high-level scientific procedures that need to be performed, such as the PCR and Cell Painting protocols examined in this work. Second, the system is provided with detailed information about the available robots' capabilities, including their possible actions, required arguments for each action, and unique operational constraints. Third, the system receives specifications for the expected output format, detailing the structure of the YAML file that will be used by the laboratory automation system (Argonne's MADSci[26] framework, building on earlier work[27]) to execute the protocol. These documents can be used as separate files when working with tools like Claude Code, which allows for better composition and organization, or they can be concatenated into a single prompt when using models *via* direct API calls to the models as we did in this work.

The YAML output format follows the MADSci framework's protocol specification, which encodes a protocol as an ordered sequence of atomic robot actions. Each step in the YAML file includes the robot to use, the action to perform, and any arguments required for that action. This structured format breaks down high-level instructions like "run PCR" into a precise sequence of robot movements, plate transfers, and device operations. The YAML structure follows a consistent pattern that allows the laboratory automation system to interpret and execute each step sequentially, with appropriate error handling and reporting opportunities between steps.

Validation of the generated YAML files is handled by the MADSci framework, which performs syntax checking and basic capability verification. MADSci examines the YAML file to ensure that all required fields are present, that the robots referenced in the protocol exist in the laboratory configuration, and that the requested actions are within each robot's capabilities. This built-in validation eliminates the need for separate validation code in our system. MADSci also checks whether the arguments provided for each action match the expected inputs for the corresponding robot. The additional, more advanced validation that we introduce, such as collision detection and object presence verification, is handled during the simulation phase, where physical execution of the protocol is tested in a virtual environment.

Robot information is organized to make laboratory setup more flexible and reusable. Each robot's capabilities, actions, arguments, and unique operational requirements are defined once and packaged together. This approach allows researchers to define laboratory-specific restrictions separately from the intrinsic properties of the robots. When setting up a new laboratory configuration, researchers can simply specify which robots are being used and derive prompts from these pre-written descriptions, then add any laboratory-specific constraints without needing to redefine the robots' capabilities. This separation makes it easier to compose new laboratory layouts by reusing existing robot definitions and only adding the minimal information needed for the specific laboratory setup being modeled.

2.3 Iterative error detection in simulation

To build the digital twin environment for our simulation we acquired Computer Aided Design (CAD) models of the robots from vendors or public repositories. These models were initially static, lacking physics and joint information needed for movement simulation. We added necessary physics properties to each model by defining collision geometries, and placing joint connections at locations that match the real robots, a process which required measuring the physical robots' joint positions and mapping them to corresponding locations in the CAD models. We also developed simulation drivers that translate commands intended for the real robots into the appropriate control signals for moving the simulated joints in Omniverse,[28] allowing the digital robots to mimic the movements of their physical counterparts. These collision and joint additions enable the simulation to be a more useful model of the real world for the downstream error detection steps without requiring substantial extra effort by the researcher.

Fig. 2 illustrates this process for the Opentrons OT-2: starting from the manufacturer-provided CAD model (A), we added rendering materials (B), collision meshes for contact detection (C), and joint definitions to enable articulated motion (D). The fully assembled simulation scene (E) is shown alongside the physical RAPID-446 laboratory (F), with instruments color-matched to highlight the correspondence between virtual and real-world placements. All instruments were fully defined with materials, collision meshes, and joints (A–D) before placing in the same configuration as the real-world placements.

Setting up the laboratory scene involved measuring the real-world laboratory space and placing the digital robots in matching positions within the simulation. A CAD model's high fidelity allows for beyond millimeter-level accuracy in robot size and placement for future research in more flexible robot control. However, this implementation focused on functional

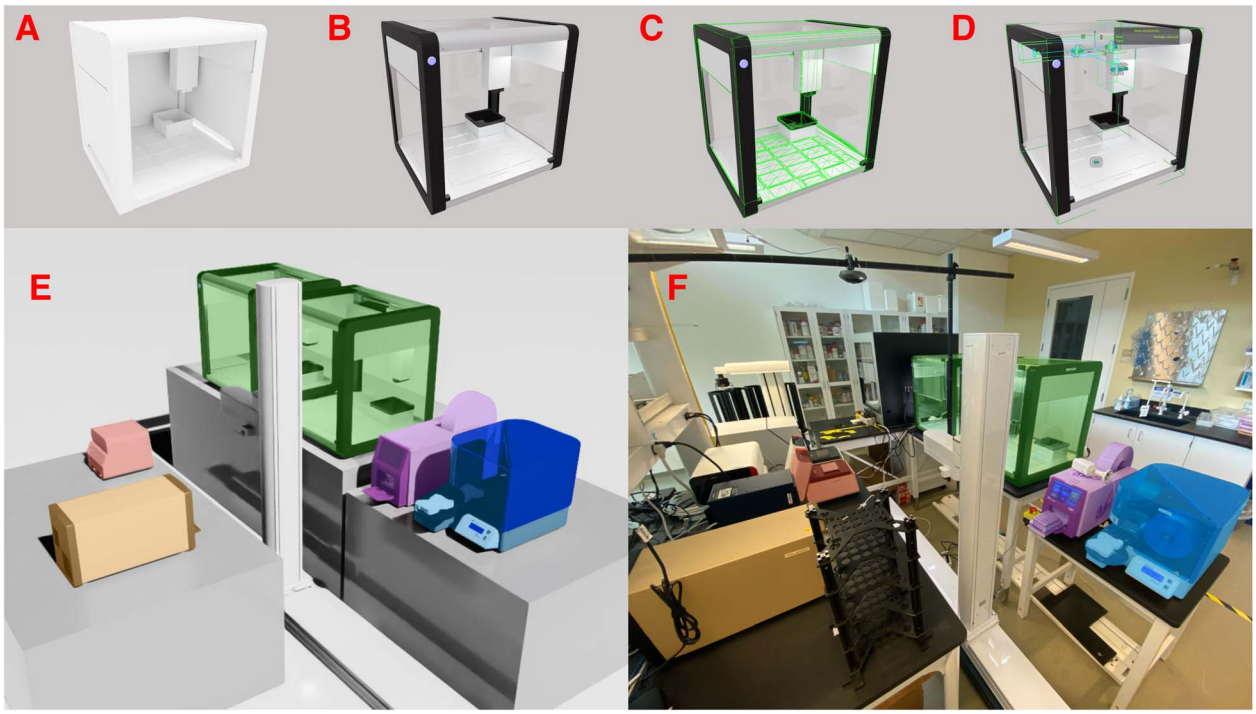

Fig. 2 Construction of the digital twin environment. (A–D) Successive stages of preparing a single instrument (Opentrons OT-2) for simulation: (A) raw CAD model as imported from the manufacturer; (B) addition of realistic materials for rendering and potential use with multimodal models; (C) collision meshes (outlined in green) added to all surfaces to enable detection of improper movements; (D) joint definitions with bounding boxes and debug visualization, enabling articulated motion of movable components. (E and F) Side-by-side comparison of the assembled simulation scene (E) and the corresponding physical laboratory, RAPID-446 (F). Instruments are color-matched between views: thermocycler (red), Hidex plate reader (orange), Opentrons OT-2 (green), peeler (blue), and sealer (purple). The PF400 microplate transfer arm (center, white tower on rail) provides access to all instruments.

accuracy rather than precise positioning as robot motions occurred only in predefined configurations. By placing the robots in a visually matching configuration, we enabled reliable collision detection during key events such as placing and retrieving plates from robots and simplified manual visual confirmation that protocol simulation was proceeding as expected.

2.3.1 Sim-to-real correspondence. The communication between MADSci and the simulation environment uses custom robot interface modules that replace MADSci's default connections to physical robots with customized network messages to the digital twin over ZeroMQ. This approach allows MADSci to operate as if it were communicating with real robots while in actuality sending commands to the simulation. The Omniverse scripts receive these messages and direct the simulated robots to perform the requested actions, then send completion or error messages back to MADSci. This communication method avoids the complexities of replicating each robot's unique network protocol, leveraging the proven MADSci capabilities.

An important aspect of our approach is that MADSci behaves identically whether controlling real or simulated robots. We run the same MADSci software used in physical laboratories, with only the robot interface modules changed to communicate with the simulation. This means that all command validation, orchestration logic, and error handling built into MADSci operates exactly as it would in a real-world setting. The timing and synchronization of robot actions occur in real-time, with messages sent to and from the simulation as they would be with physical robots. This consistency gives confidence that protocols working correctly in simulation will behave similarly when transferred to the real laboratory, as the only difference is the replacement of physical robots with their simulated counterparts.

2.3.2 Error detection feedback. The simulation detects several types of errors through physics-based collision detection. We configure the system to ignore expected collisions, such as when a robot gripper intentionally contacts a plate that it is meant to pick up, while monitoring solely for unexpected collisions between robots or between materials and the environment. When an unintended collision occurs, the simulation reports it to MADSci with an English-language description of the problem (identifying which objects collided by name). This error reporting causes the MADSci orchestrator to terminate the experiment execution, just as it would if a similar error occurred in the physical laboratory. The complete execution log up to the point of failure can be captured, providing a record of all steps successfully completed before the error plus context about what was happening when the collision was detected.

Beyond collision detection, the simulation checks for object presence and validates command execution. A ray-casting system verifies whether plates or other materials are present when robots attempt to interact with them. If a robot tries to

pick up a plate that is not at the expected location, the system detects this absence and reports an error. Similarly, if a robot attempts to place a plate when it is not currently holding one, this inconsistency triggers an error. The simulation also validates whether commands are executable given the robots' physical constraints, such as detecting when a requested position is outside a robot's reach or when joint limitations would prevent a specified movement. All errors are reported with a useful English-language description of the problem and any relevant details from the simulation, allowing the language model to more easily identify and reason over the problem. These runtime checks help identify issues that might not be apparent when manually reviewing a protocol but would cause failures during physical execution.

2.3.3 Iteration process. When the simulation detects an error, we append the error as a new message in the conversation with the language model, which is sufficient for guiding the model to think about and fix the issue. Models exhibit different patterns when correcting errors in protocols, ranging from immediately identifying and fixing the issue, to misdiagnosing the problem and introducing ineffective solutions or even new errors. Through the use of the simulation, all of these possible behavior patterns are handled by iteratively prompting the model with detected errors from each attempt, allowing the model to explore and learn from its mistakes. This is done until the model converges to a protocol without any errors.

A successful result indicates that the physical aspects of the protocol should execute without issues in the real laboratory as the robots' movements, plate transfers, and device operations have all been verified in the digital twin. However, it's important to note that simulation success does not guarantee scientific accuracy. The simulation confirms physical executability but cannot verify that the protocol will achieve the intended scientific outcome, as we do not simulate liquid physics, chemical reactions, or biological processes. After a protocol passes simulation validation, it undergoes human review to verify scientific accuracy. In our work, a researcher still examines the final protocol to confirm that it performs the requested scientific operations in the correct sequence with appropriate parameters. This manual review step ensures that in the process of fixing physical execution issues the model did not introduced changes that would compromise the experiment's scientific validity.

2.4 Real world execution

We selected two representative experimental workflows to evaluate PRISM capabilities: a PCR amplification protocol and a Cell Painting assay. We choose these two workflows because they (i) encompass a range of laboratory operations, reagent types, and protocol structures; (ii) are widely used in biological research; (iii) are structurally distinct; and (iv) capture the types of reasoning challenges that PRISM is designed to address. PCR provides a compact, highly standardized molecular biology workflow, while Cell Painting represents a multi-step, multi-reagent imaging assay with substantially higher procedural complexity.

We selected PCR amplification as a first benchmark task because it is a canonical, well-defined molecular biology procedure with a clear, measurable outcome. Benchmarking against PCR allows direct comparison to prior work on protocol generation and provides a reproducible baseline for evaluating end-to-end execution on a robotic system. In a typical PCR workflow, a reaction mixture containing a DNA template, primers, nucleotides, and polymerase is assembled and subjected to a series of temperature cycles to amplify the target DNA. Validation is commonly performed by gel electrophoresis. However, our automated laboratory does not include gel-handling capabilities. To enable fully autonomous execution, we used the Luna qPCR Master Mix (New England Biolabs), which incorporates an intercalating dye allowing real-time

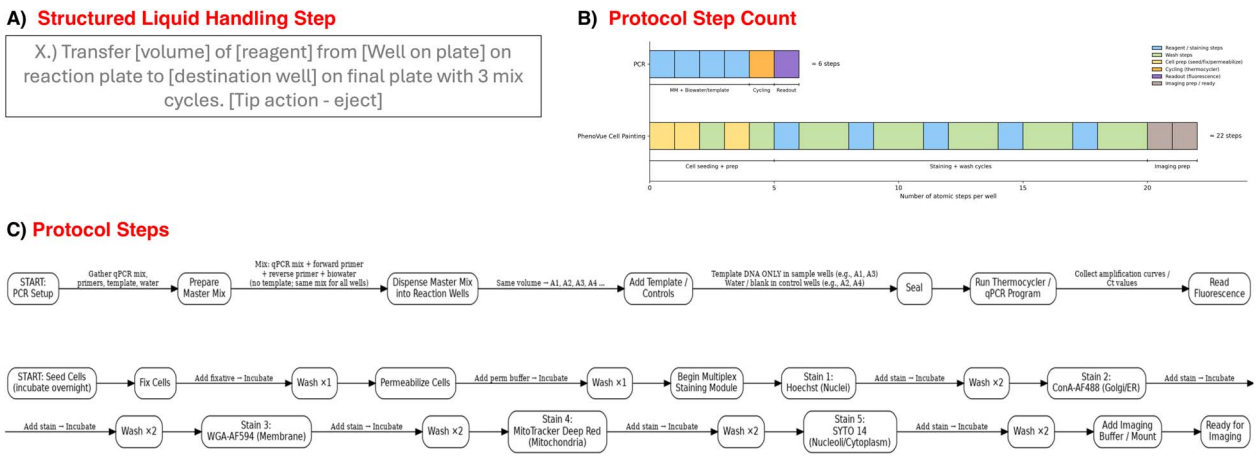

Fig. 3 Comparison of protocol complexity for Luna qPCR and PhenoVue Cell Painting assays. (A) Structured liquid-handling step format that all generated protocols must follow. (B) Number of atomic steps required per well/reaction for each assay, coloured by operation type. Cell painting requires substantially more reagent additions and wash cycles than qPCR, and is highly order-dependent (*e.g.*, stain–wash cycles must be performed sequentially), whereas reagent addition order does not matter in qPCR. (C) Schematic step-by-step workflows for qPCR (top) and Cell Painting (bottom), highlighting the increased length, repetition, and ordering constraints of the Cell Painting protocol.

fluorescence readout. This adaptation provides an automated, quantitative signal equivalent to confirming the presence of an amplified product, without requiring any manual steps.

The protocol generated by PRISM was executed end-to-end on our automated laboratory platform, which integrates an Opentrons OT-2 liquid handler,[29] a PF400 robotic arm,[30] a Hidex plate reader,[31] and an Azenta plate sealer[32] and peeler.[33] This experiment demonstrates that PRISM can produce robot-executable protocols that run to completion with no human intervention.

To assess generality beyond PCR, we also applied PRISM to generate a protocol for a Cell Painting assay, a standard morphological profiling workflow widely used in high-content imaging screens. In a typical Cell Painting assay (see Fig. 3), adherent cells are seeded, fixed, permeabilized, and stained using a panel of fluorescent dyes that highlight cellular structures such as nuclei, endoplasmic reticulum, Golgi, and mitochondria. The stained cells are then imaged using a high-content screening microscope to extract morphological features. Cell painting exercises protocol-generation capabilities that PCR does not: multi-step washing and incubation, sequence-dependent operations, complex reagent handling, and long branching workflows. These characteristics make it an ideal test of PRISM's ability to handle procedurally complex, multi-module protocols.

We evaluated the Cell Painting protocol produced by PRISM *in silico*. A domain expert reviewed the generated steps for biological correctness and internal consistency. Although the core Cell Painting steps (fixation, permeabilization, staining, and washing) can be emulated on our OT-2 through controlled pipetting, our current workcell does not include an integrated automated plate washer. Because Cell Painting typically involves numerous gentle wash cycles optimized for preserving adherent cell monolayers, dedicated plate washers are commonly used in high-content screening workflows. For this reason and because our platform lacks a high-content imaging system, we validated the Cell Painting protocol *in silico* rather than executing it physically.

3 Results and discussion

We present our results in an order that reflects the three-stage pipeline of the PRISM framework illustrated in Fig. 1. Section 3.1 evaluates stage 1, where we assess different approaches to protocol planning that convert high-level experimental intent into structured English instructions. This includes comparing multi-agent *versus* single-agent frameworks and constrained *versus* open-ended prompting strategies. Section 3.2 corresponds to stage 2, the protocol generation and validation loop, where structured steps are translated into robot-executable YAML formats and iteratively refined through simulation-based error detection. We also present an ablation study demonstrating the necessity of simulation for achieving physical correctness. Finally, Section 3.3 presents stage 3, where simulation-validated protocols are executed on our automated laboratory platform to demonstrate end-to-end functionality and biological validity.

3.1 Protocol planning

We begin our evaluation by focusing on the initial phase of the PRISM framework: the translation of scientific intent into structured experimental plans. This stage serves as the foundational logic layer; errors introduced here propagate downstream, rendering even syntactically perfect robotic code scientifically invalid. To assess this capability, we benchmark performance across two dimensions: the architectural framework (comparing single-agent reasoning against the proposed multi-agent system) and the prompt constraints (evaluating adaptability under both rigid and open-ended instructions).

3.1.1 Experimental setup and evaluation criteria. We evaluated five LLMs (GPT-5, Claude Opus 4.1, Claude Sonnet 4.5, Gemini 2.5 Pro, and Gemini 2.5 Flash) on their ability to generate structured liquid handling steps in plain English for the PCR and Cell Painting workflows described in Section 2.4.

For PCR, each model was tested under both the multi-agent and single-agent frameworks using both the constrained and open-ended prompting paradigms. Performance was assessed based on (1) the number of correction iterations required to refine the generated protocol and (2) the types of logical, formatting, or physical errors present in the generated steps. We allowed at most three correction iterations; if a model failed to produce a valid protocol within this limit, the attempt was recorded as non-convergent. The PCR results were reviewed by a biologist and experimentally validated on our robotic platform.

We evaluated each generated protocol for both structural correctness and biological feasibility. Under the constrained prompting setup, models received fixed experimental parameters, reagent mappings, and well layouts designed to minimize ambiguity, ensuring that success reflected precise adherence to given instructions rather than creative inference. Performance was assessed across six predefined criteria: (1) correct source and destination well assignments, (2) correct reagent volumes, (3) inclusion of exactly three thermocycling steps, (4) adherence to the required step format, (5) correct reagent use for test and control wells, and (6) appropriate pipette selection.

In contrast, the open-ended prompting setup provided only high-level experimental goals without fixed reagent mappings or per-component volumes, while still specifying essential lab constraints such as the OT-2 deck layout (*e.g.*, slot assignments for the reaction and destination plates) and a target total reaction volume. Therefore, models were required to infer compatible volume splits, well layouts, and transfer sequences that satisfied these physical and biological constraints. Evaluation in this setting focused on eight criteria that capture reasoning flexibility and practical feasibility: (1) practical well mapping, (2) adequate spacing between test and control wells to avoid cross-contamination, (3) biologically correct volume calculations, (4) inclusion of exactly three thermocycling steps, (5) correct step formatting, (6) correct reagent logic for test and control wells, (7) proper and consistent pipette use, and (8) compliance with minimum volume and plate capacity limits.

We provide the full prompt templates and model instructions for both the constrained and open-ended paradigms in

the SI. Quantitative accuracy across the criteria listed above was measured using the F_1 score, which captures both the precision and completeness of the generated protocol relative to the ground-truth reference. The F_1 score was calculated as,

$$F_1 = \frac{2 \times \mathrm{TP}}{2 \times \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

where TP, FP, and FN represent the number of correct, additional, and missing steps, respectively. The F_1 score accounts for additions, deletions, and errors in various ground truth criteria chosen for each prompt paradigms.

3.1.2 Use case 1: visual exploration of PRISM capabilities. We first evaluated PRISM in our Rapid Prototyping Laboratory (RPL) on a set of plate-pattern tasks to demonstrate its ability to generate correct OT-2 protocols outside a biological laboratory. Specifically, we validated PRISM's spatial planning capabilities by replicating the geometric pattern tasks previously demonstrated by Boiko *et al.*,[34] confirming that our system can successfully interpret high-level visual descriptions into robot-executable protocols. These non-biological demonstrations show that PRISM can infer well locations, compute volumes, assign reagent sources, and output robot-executable liquid-handling instructions based solely on a natural-language goal. Fig. 5 illustrates the four patterns used: (A) a 3×3 colored block in the upper-left corner, (B) a diagonal starting from the lower-left corner of the plate, (C) an alternating-row fill pattern, and (D) a rainbow checkerboard created by applying a color gradient to every other well. These tasks highlight PRISM's capacity to generalize beyond assay-specific workflows and generate protocols for arbitrary user-defined objectives. Using only an English-language description of which wells to fill and how to color each well, PRISM determines exact well locations and ratios of input liquids to complete the task.

3.1.3 Use case 2: PCR protocol generation. Next, we evaluated PRISM in RAPID-446, our BSL-1 laboratory, to show that PRISM generalizes to different laboratory configurations and different instruments. RAPID-446 contains additional robots in a different physical layout to that of RPL, allowing us to demonstrate that PRISM is not restricted to the RPL configuration that we used for development.

Fig. 4a and b summarize per-model F_1 accuracy for PCR across the two frameworks, while Table 1 reports the number of

Ground Truth Criteria			GPT 5	Claude Opus 4.1	Claude Sonnet 4.5	Gemini 2.5 Pro	Gemini 2.5 Flash
Multi-Agent workflow	Constrained	Well Assignment	✓	✓	✓	✓	✓
		Reagent Volumes	✓	✓	✓	✓	×
		Exactly 3 Thermocycling Steps	✓	×	×	×	✓
		Step Format	✓	×	×	×	×
		Reagent Logic	✓	✓	✓	✓	×
		Practical Pipette Use	✓	✓	✓	✓	✓
		F1 Score	**1.0**	**0.80**	**0.80**	**0.80**	**0.67**
	Open -Ended	Practical Well Assignment	✓	✓	✓	✓	✓
		Adequate Well Spacing	✓	✓	✓	✓	×
		Correct Volume Calculations	✓	×	×	×	×
		Exactly 3 Thermocycling Steps	✓	×	×	×	×
		Step Format	✓	×	×	×	×
		Reagent Logic	✓	✓	✓	×	×
		Practical Pipette Use	✓	✓	✓	×	×
		Volume and capacity compliance	✓	✓	✓	×	×
		F1 Score	**1.0**	**0.77**	**0.77**	**0.40**	**0.22**

(a) Multi-agent

Ground Truth Criteria			GPT 5	Claude Opus 4.1	Claude Sonnet 4.5	Gemini 2.5 Pro	Gemini 2.5 Flash
Single Agent Workflow	Constrained	Well Assignment	✓	✓	✓	✓	✓
		Reagent Volumes	✓	✓	✓	✓	✓
		Exactly 3 Thermocycling Steps	✓	✓	×	✓	×
		Step Format	✓	×	×	×	×
		Reagent Logic	✓	✓	✓	✓	✓
		Pipette Use	✓	✓	✓	✓	✓
		F1 Score	**1.00**	**0.91**	**0.80**	**0.91**	**0.80**
	Open -Ended	Practical Well Assignment	✓	✓	✓	✓	✓
		Adequate Well Spacing	✓	✓	✓	✓	✓
		Correct Volume Calculations	×	×	×	✓	✓
		Exactly 3 Thermocycling Steps	×	✓	×	×	×
		Step Format	✓	×	×	×	×
		Reagent Logic	×	×	×	×	×
		Pipette Use	×	×	×	×	×
		Volume and capacity compliance	✓	✓	✓	✓	✓
		F1 Score	**0.67**	**0.67**	**0.55**	**0.67**	**0.67**

(b) Single-agent

Fig. 4 Comparison of multi-agent and single-agent workflow performance on Luna qPCR protocol generation. Each panel summarizes per-model accuracy across all ground-truth evaluation criteria for five LLMs: GPT-5, Claude Opus 4.1, Claude Sonnet 4.5, Gemini 2.5 Pro, and Gemini 2.5 Flash. ✓ and ✗ indicate whether each criterion was met or violated. Color coding denotes the type of deviation from the ground truth: yellow = extra action (false positive), red = missing action (false negative). These errors contribute to the F_1-score reported in the bottom row, which quantifies overall protocol accuracy. Panel (a) shows the multi-agent workflow (WebSurfer → planner → critique → validator), and panel (b) shows a single agent workflow without modular decomposition of tasks. The comparison highlights that structured multi-agent reasoning achieves higher correctness and more reliable protocol synthesis than monolithic reasoning generation.

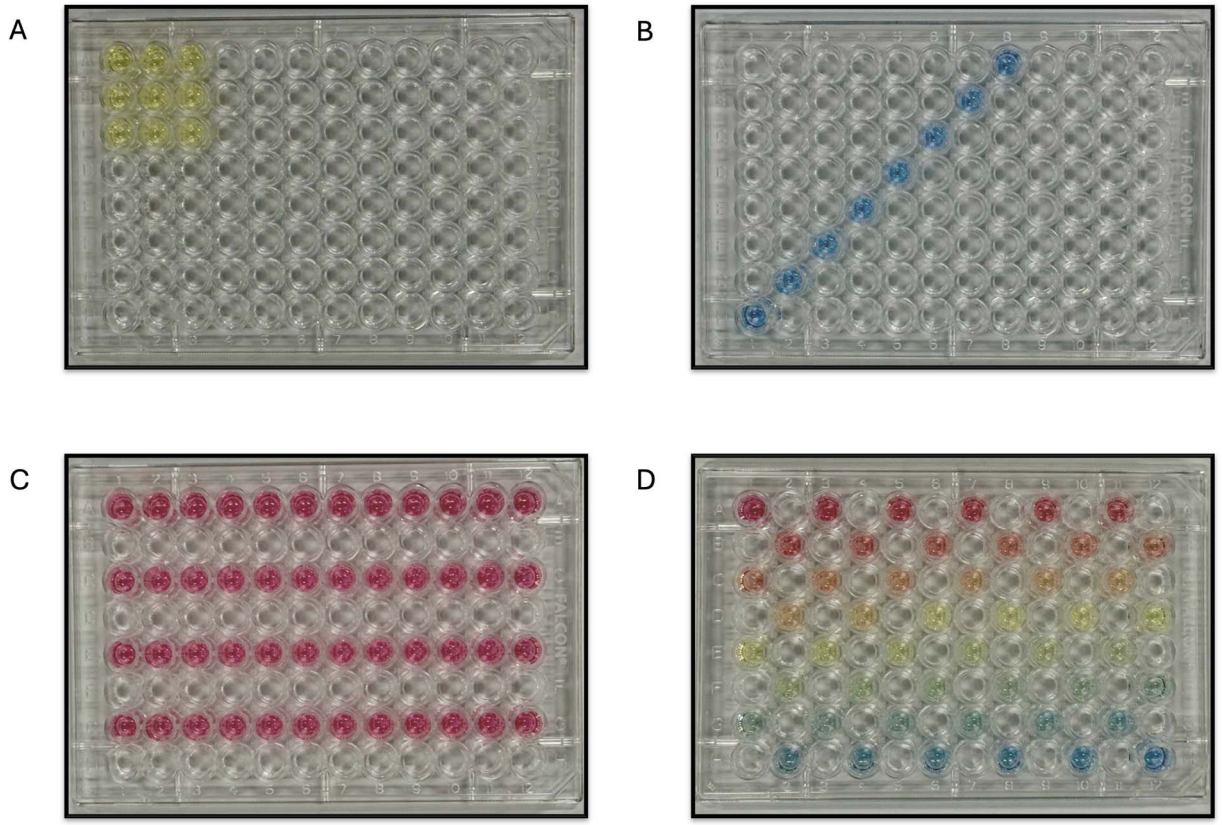

Fig. 5 Visualization-focused pattern-generation tasks used to evaluate PRISM. (A) A 3 × 3 coloured square in the upper-left corner. (B) A diagonal pattern beginning at the lower-left corner (H1). (C) An alternating-row fill pattern. (D) A rainbow checkerboard combining a red-yellow-blue gradient with a parity-based fill rule.

Table 1 Luna qPCR protocol results showing the number of refinement iterations (1–3) required for each model to achieve convergence under the multi-agent and single-agent frameworks across constrained and open-ended prompting. ✓ indicates that the model produced a correct protocol on the first attempt without requiring refinement; integers (1–3) denote the number of refinement cycles needed before convergence; and † indicates non-convergence within three iterations

	Multi-agent		Single-agent	
Model	Constrained	Open-ended	Constrained	Open-ended
GPT-5	✓	✓	✓	2
Claude Opus 4.1	2	†	†	†
Claude Sonnet 4.5	†	†	†	†
Gemini 2.5 Pro	†	†	†	†
Gemini 2.5 Flash	†	†	†	†

correction iterations required for each model to converge. Together, these analyses capture both the accuracy of protocol generation and the adaptability of each framework under iterative refinement.

Under constrained prompting, GPT-5 achieved perfect F_1 scores for both the multi-agent and single-agent frameworks, generating a correct protocol in the first attempt. Under open-ended prompting, GPT-5 initially showed minor inconsistencies in volume calculations, step formatting, number of thermocycling steps, and transfer of reagents but converged to fully correct protocols within three refinement iterations.

In contrast, the Claude and Gemini models did not converge reliably across iterations, with several configurations showing degraded performance over time. For example, Gemini 2.5 Flash (reasoning with constrained prompts), Claude Opus 4.1 (reasoning with fixed constraints), and Claude Opus 4.1 (open-ended multi-agent framework) often added redundant steps, scaled volumes to impractical levels, or altered previously correct actions. These models also struggled to maintain the required step format and failed to produce exactly three thermocycling steps under constrained prompting. When open-ended prompts were used they deviated further across evaluation criteria particularly in maintaining biologically valid volume or concentration calculations compatible with the robotic platform. Overall, the multi-agent framework still achieved higher accuracy and faster convergence for most non-GPT models, largely because its structured feedback helped recover from format and logic inconsistencies that single reasoning chains could not self-correct. However, they still had minor issues in step-format and therefore we say they did not fully converge in Table 1.

These results indicate that model performance is shaped not only by parameter capacity but also by reasoning dynamics: the GPT model exhibits sustained contextual reasoning that allows it to manage experimental constraints implicitly, whereas

models like Gemini and Claude, which favor faster response generation, often lose context or respond without sufficient internal deliberation, requiring more explicit structure and corrective feedback to achieve valid protocols.

3.1.4 Use case 3: Cell Painting. To evaluate performance on more complex experimental workflows, we applied both the multi-agent and single-agent frameworks to generate a Cell Painting protocol. This evaluation was not benchmarked across prompt paradigms, as the goal was to test whether each framework could handle a longer, multi-step procedure rather than optimize specific prompting conditions. The generated protocols were reviewed by a biologist to confirm that they were biologically sound and physically feasible on our automated platform. In this setting, reasoning models often failed to maintain coherent step structure as the number of actions increased, frequently omitting reagents or misordering procedural steps. In contrast, the multi-agent framework produced a consistent and logically organized protocol that captured the necessary sequence of reagent additions and washing steps. The division of tasks across specialized agents helped maintain structure and correctness as procedural complexity increased, underscoring the importance of modular, feedback-driven design for planning larger, multi-phase experiments.

Overall, these results show that structured prompting and modular task decomposition play complementary roles in generating correct and physically executable experimental protocols. High-capacity models such as GPT-5 perform reliably across both frameworks and prompt types, while smaller models benefit substantially from the structured feedback of the multi-agent system and the explicit guidance of constrained prompts. The multi-agent framework offers a clear advantage when protocols become longer or more detailed, enabling consistent correction of logical and formatting errors that reasoning models alone struggle to resolve. Together, these findings demonstrate accurate and adaptable protocol planning and provides a foundation for the next stage of translating the structural steps into fully executable robotic workflows.

Ground Truth	GPT-5	Opus 4.1	Sonnet 4.5	Gemini 2.5 Pro	Gemini 2.5 Flash	Gemini 2.5 Flash-Lite
1. ot2 run_protocol	✔	✔	✔	✔	✔	✔
2. transfer → exchange	✔	✔	✔	✔	✔	✔
3. transfer → sealer	✔	✔	✔	✔	✔	✔
4. seal	✔	✔	✔	✔	✔	✔
5. open thermocycler	✔					
6. transfer → exchange	✔	✔	✔	✔	✔	✔
7. transfer → thermocycler	✔	✔	✔	✔	✔	✔
			5	5	5	5
8. close thermocycler	✔	✔	✔	✔	✔	✔
9. run thermocycler	✔	✔	✔	✔	✔	✔
10. open thermocycler	✔	✔	✔	✔	✔	✔
11. transfer → exchange	✔	✔	✔	✔	✔	✔
12. transfer → peeler	✔	✔	✔	✔	✔	
13. peel	✔	✔	✔	✔	✔	
	+transfer		+transfer			
14. open hidex	✔					
15. transfer → hidex	✔	✔	✔	✔	✔	✔
			14	14	14	14
16. close hidex	✔	✔	✔	✔	✔	✔
17. run hidex	✔	✔	✔	✔	✔	✔
18. open hidex	✔	✔	✔	✔	✔	✔
F1 Score	1.00	0.94	0.88	0.88	0.88	0.82

Missing action (False Negative) Extra action (False Positive) Benign insertion

Fig. 6 Initial generated protocols *vs.* ground truth for a PCR workflow. Each column shows the first protocol generated by the specified reasoning model compared against the correct sequence of actions. ✓ indicates correct action placement. Red cells indicate missing actions, yellow cells indicate inserted actions (the number corresponds to correct position of the action), and blue cells indicate benign modifications where the model added unnecessary but non-harmful intermediate steps. GPT-5 achieved perfect initial generation, while other models exhibited a common failure pattern: omitting the open commands required before transferring plates into the thermocycler (step 5) and plate reader (step 14).

3.2 Protocol generation/validation

To evaluate the initial protocol generation capabilities of each model, we assessed their ability to produce a complete, executable PCR workflow from structured English instructions in a single attempt. This zero-shot evaluation provides a baseline measure of each model's understanding of robotic constraints, sequencing requirements, and the physical limitations of the laboratory environment. Fig. 6 presents a comparative analysis of the first protocol generated by each of the six tested reasoning models against the ground truth workflow.

The analysis reveals a striking pattern across the models: five of the six tested systems (Claude Opus 4.1, Claude Sonnet 4.5, Gemini 2.5 Pro, Gemini 2.5 Flash, and Gemini 2.5 Flash-Lite) all omitted the open commands required before placing plates into the thermocycler and plate reader. This error type represents a physical impossibility: attempting to insert a plate into a closed device would cause a failed experiment and potential equipment damage in the physical laboratory. The error pattern demonstrates a consistent gap in long-horizon planning: while models successfully generated most of the basic protocol structure, they failed to anticipate that certain devices require extra operations before interaction, a constraint that differs from the typical usage pattern of the other laboratory robots.

Notably, GPT-5 achieved perfect initial generation with an F_1 score of 1.0, producing a physically executable protocol without any errors on the first attempt. The performance gradient across models, reflected in declining F_1 scores from GPT-5 (1.0) through Claude Opus 4.1 (0.94) down to Gemini 2.5 Flash-Lite (0.82), shows quantitatively how model capability directly impacts initial generation quality. Less capable models exhibited higher rates of missing actions and a failure to fully capture the basic protocol structure.

These zero-shot results establish the fundamental challenge that the PRISM framework addresses: even state-of-the-art language models often produce physically infeasible protocols when translating high-level scientific intent into executable robotic instructions. The errors shown in Fig. 6 would cause

failures if executed directly on physical hardware, motivating the core function of PRISM's simulation-based refinement loop to iteratively detect and correct these flaws before real-world execution.

3.2.1 Iterative refinement through simulation feedback. All models except GPT-5 required further iterations with simulation feedback to produce valid protocols. The simulation environment detected physical impossibilities, such as attempting to place plates into closed devices, and returned error messages describing the specific collision or constraint violation. Models demonstrated the ability to interpret these error messages, identify the problematic steps, and propose corrections, though the efficiency and pattern of error correction varied by model.

The medium-capability models, Claude Sonnet 4.5 and Gemini 2.5 Flash, successfully interpreted simulation error messages and correctly diagnosed the root causes of failures. However, these models exhibited a localized correction pattern: when the simulation flagged a specific instance of an error (*e.g.*, attempting to place a plate into a closed device at a specific step, as shown in Fig. 7), the model would fix only that particular occurrence. Subsequent simulation runs would then generate new error messages for other instances where the same mistake had been made, requiring additional iterations to systematically eliminate all occurrences of the error pattern. This behavior resulted in three iterations for both Claude Sonnet 4.5 and Gemini 2.5 Flash to achieve valid protocols, as shown in Table 2.

In contrast, the stronger models, Claude Opus 4.1 and Gemini 2.5 Pro, demonstrated global pattern recognition. When provided with an error message about a single instance of a mistake, these models recognized that they had made the same error in multiple locations throughout the protocol and corrected all instances simultaneously in a single iteration. This capability to generalize from one error instance to identify and fix a systematic problem pattern enabled these models to achieve valid protocols in just two iterations.

The weakest tested model, Gemini 2.5 Flash-Lite, exhibited substantially less capable error diagnosis behavior. Rather than immediately identifying the correct root cause, this model incorrectly handled the initial error and introduced new errors in the process. Only after accumulating additional context from further error messages did the model converge on a physically possible solution. This trial-and-error approach resulted in three iterations before achieving a solution that resolved the physics-based errors detected by the simulation. However, despite producing a physically executable protocol that passed all simulation checks, the model still failed to include the peel command, a scientifically necessary step for correct results from the plate reader that our physics simulation did not verify. When the human researcher provided a manually formatted error message indicating the missing operation, Gemini 2.5 Flash-Lite successfully incorporated a correction in its final iteration. This complete refinement process resulted in four total iterations before achieving both a physically valid and scientifically complete protocol, demonstrating that while simulation-based feedback substantially aids weaker models in achieving physical correctness, it does not eliminate the need for scientific review of the final protocol.

Table 2 Number of simulation iterations required for each model to produce a physically valid PCR protocol in simulation. GPT-5 generated a correct protocol without any errors in the first simulation run, while weaker models required progressively more attempts to identify and correct all errors. Gemini 2.5 Flash-Lite passed all simulation checks in 3 iterations, however it failed to include the peel command, a scientifically necessary step for correct results from the plate reader that our physics simulation did not verify

Model	Total iterations
GPT-5	1
Claude Opus 4.1	2
Gemini 2.5 Pro	2
Claude Sonnet 4.5	3
Gemini 2.5 Flash	3
Gemini 2.5 Flash-Lite	3 (failed scientific requirements)

To assess the generalizability of these results beyond the PCR experiment, we also evaluated all models on a Cell Painting protocol. In this case, every model produced a correct protocol on the first attempt, requiring zero iterations for validation. This universal success was due to the fact that the majority of the complexity in the Cell Painting protocol lies in the liquid handling steps covered thoroughly by the planning phase (Fig. 1, stage 1) as shown in Fig. 3, as well as the reduced robotic variety of the Cell Painting workflow, which lacked the intricate coordination requirements between robots that induced most of the model errors with the PCR protocol. Despite this universal first-attempt success, minor differences in model behavior remained observable. Each model independently selected its own interpretation of room temperature, with values varying between 22 and 25 degrees Celsius across different models. Additionally, when prompted to seal a plate at 4 degrees Celsius for an indefinite storage period, all models except GPT-5 chose to seal the plate without placing it in the incubator, leaving a note for the human researcher to configure storage as desired. This approach reflected a conservative

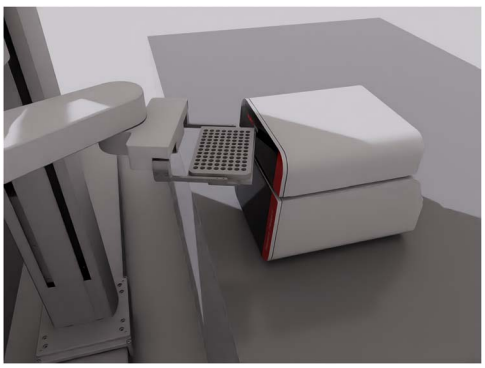

Fig. 7 Visualization of a plate collision error in the simulation environment. The robot arm attempts to insert a plate into the thermocycler while the device lid remains closed, representing a physical impossibility. This error occurred when models omitted the required open command before plate insertion.

interpretation of the input documentation, which did not describe indefinite-duration incubation as a supported capability. GPT-5 opted to place the sealed plate in the incubator at the required temperature for a specified duration of 24 hours, accompanied by a note for the researcher to adjust the duration as needed.

3.2.2 Limitations of self-correction without simulation. To evaluate the value that simulation-based validation provides beyond language model self-critique, we conducted an ablation study where the simulation environment was disabled. This experiment tested whether a reasoning model, when prompted to identify and fix its own errors, could achieve the same level of physical correctness as the complete PRISM framework with simulation feedback. For this work, we designed a more challenging scenario to stress-test the model's abstract reasoning capabilities: a protocol that processes two plates sequentially through the PCR workflow, creating a conflict where both plates would need to occupy the plate reader simultaneously, a physical impossibility. This scenario simulates a realistic failure mode where the upstream protocol planning phase produces a logically coherent but physically infeasible plan.

We provided Claude Sonnet 4.5 with this flawed plan and asked it to generate the corresponding YAML protocol. The initial generation contained multiple errors: the model forgot to include open commands for both the thermocycler and plate reader before attempting to insert plates, and it produced mismatched plate transfer orientations that would cause collisions. Additionally, the protocol's terminal state had both plates in the plate reader simultaneously, confirming that the model had not detected the physical impossibility during generation.

When prompted to review its work and identify any mistakes, the model successfully recognized the high-level logical conflict: two plates could not occupy the same location. It proposed a solution to move the first plate to an exchange deck before processing the second plate, demonstrating capable logical reasoning about spatial conflicts. However, this modification introduced a new problem. The protocol required use of the exchange deck as an intermediate transfer location for the second plate's movements, and with the first plate now occupying this position, the exchange deck was blocked. Upon a second self-critique request, the model recognized this emergent conflict and revised the plan again, orchestrating movements of the first plate back to its original position in the OT-2 to free both the plate reader and the exchange deck.

After this sequence of corrections, the model performed a final self-review and declared the protocol to be correct and ready for execution. However, multiple physical impossibilities remained undetected. The protocol still attempted to place plates into closed devices, the same error pattern shown in Fig. 6 that all models except GPT-5 produced in their initial generations. Additionally, the mismatched plate transfer orientations persisted (Fig. 8), which would cause collision failures during execution.

In the complete PRISM framework, this protocol that the model declared "correct" would have failed when executed in the simulation environment. The grounded, physics-based error messages provide a fundamentally different form of validation than self-critique, enabling the model to identify and correct errors that it could not detect through tool-unassisted reasoning.

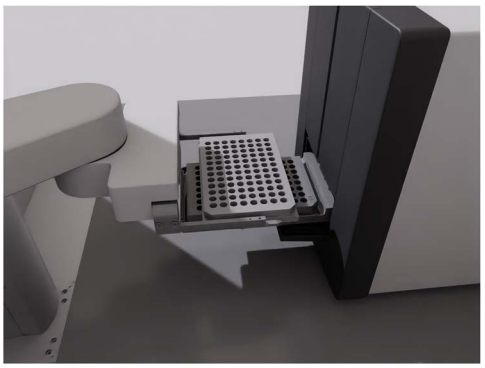

Fig. 8 Visualization of a plate orientation mismatch error in the simulation environment. The plate is incorrectly oriented during a transfer operation, resulting in a spatial configuration that would cause a collision with the robot arm or device during execution. Despite multiple rounds of self-correction where the model declared the protocol ready for execution, this type of subtle geometric constraint violation remained undetected, demonstrating the limitations of purely text-based reasoning for validating physical feasibility.

This ablation study reveals the limitations of language model self-correction for protocol refinement. The model demonstrated some sophisticated logical reasoning, successfully identifying and resolving high-level spatial conflicts through multiple iterations of abstract problem-solving. However, given the complexity of the multi-step coordination task and the subtle nature of certain physical constraints, text-based reasoning alone was insufficient to achieve a physically executable outcome.

This result establishes that the simulation component of PRISM provides an valuable validation layer. While language models possess remarkable reasoning capabilities and can identify many types of errors through self-critique, the combination of task complexity and the need to verify physical feasibility requires grounding in a physics-based environment. The simulation complements rather than replaces model reasoning, providing the physical validation that tool-unassisted reasoning cannot reliably achieve for complex, multi-step robotic protocols.

3.3 Real world experimental validation

We executed a PCR protocol generated by PRISM, as described in Section 3.1.3, on our automated laboratory platform. We used the Luna qPCR master mix, which enables fluorescence-based validation in place of gel electrophoresis, allowing fully automated execution. As shown in Fig. 9, test wells (outlined in red) produced strong amplification signals, while control wells (outlined in black) remained at baseline fluorescence. The measured RFU values were consistent with those obtained when the same PCR assay was performed manually by a scientist, indicating that PRISM-generated protocols produce biologically valid results comparable to conventional hand-executed experiments.

To evaluate the generality of PRISM beyond PCR, we used the framework to generate a Cell Painting assay protocol, as described in Section 3.1.4. PRISM successfully produced

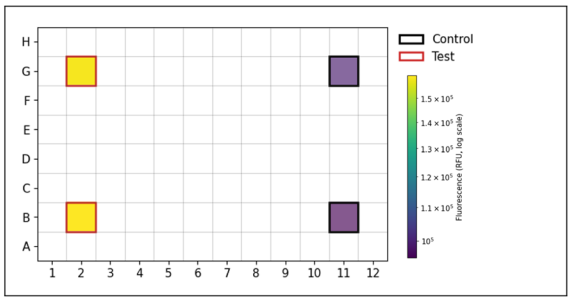

Fig. 9 PCR plate fluorescence readout from PRISM-executed protocol. Test wells (red outline) show amplified signal, while control wells (black outline) remain at baseline. Fluorescence is shown in relative fluorescence units (RFU) on a log scale.

a detailed set of robotic instructions covering reagent handling, plate setup, and imaging preparation. These instructions were validated *in silico* and by a domain expert.

Together, these results confirm that PRISM can deliver end-to-end executable protocols within an automated laboratory and can also generalize beyond single-step molecular assays to more complex, multi-step biological workflows.

4 Conclusions

We have presented PRISM, a framework that addresses the challenge of translating scientific hypotheses into error-free robotic instructions by coupling LLM-based planning systems with simulation-based validation. This integration ensures that protocols are vetted for physical reliability, enabling the application of generative AI within laboratory settings in which reliability concerns would otherwise prevent its use. We summarize in the following the system's performance across our key benchmarks, synthesize the insights gained regarding agentic reasoning, and outline the remaining challenges for fully self-driving laboratories.

4.1 System performance and contributions

We have demonstrated in this work the successful application of LLMs to automated protocol generation for laboratory research, establishing a framework that bridges the substantial gap between high-level experimental intent and executable robotic instructions. PRISM has shown compatibility with existing automated laboratory hardware by leveraging the MADSci platform, indicating that the approach does not require specialized or custom-built robotic systems to function effectively. Through the integration of simulation-based refinement *via* NVIDIA Omniverse and Isaac Sim, we have validated an approach that allows protocols to be iteratively improved before physical execution, reducing the risk of equipment damage and experimental failures.

We also reported on the results of a quantitative performance comparison across multiple LLMs. This study revealed distinct capabilities and limitations of different reasoning models in the context of laboratory automation. The system's effectiveness has been explicitly tied to real-world validation through successful PCR execution in our automated laboratory, while its generalizability has been demonstrated through *in silico* generation of a more complex Cell Painting protocol.

4.2 Key findings and insights

The protocol planning evaluation revealed fundamental differences in how language models approach the decomposition of experimental procedures into executable steps. Multi-agent workflows demonstrated superior performance by enforcing explicit separation of concerns: retrieval of procedural knowledge, spatial reasoning about deck layout and labware positioning, step-by-step formatting with physical constraints, and systematic validation of completeness and logical flow. This structured approach reduced false positives (unnecessary actions) and false negatives (missing critical steps) compared to single-agent reasoning, which tended to produce protocols that were either overly verbose or missing essential details. Notably, the iterative refinement mechanism—where the critique and validator agents provide feedback to the planner—proved essential for achieving convergence, with most models requiring 2 or 3 refinement cycles to reach acceptable accuracy levels. The comparison between constrained and open-ended prompting revealed that while constrained prompts improved initial generation quality by providing explicit guidance, open-ended prompts tested the models' ability to reason independently about experimental requirements, volumes, and transfer sequences. This finding suggests that protocol planning performance depends not only on model capability but also on the degree of structural scaffolding provided through prompt design and agent specialization.

Our evaluation revealed model-specific strengths for different aspects of protocol generation, with variations in both initial generation quality and error correction capabilities. The effectiveness of iterative refinement in achieving valid protocols was demonstrated across all tested models, though the improvement per-iteration varied significantly based on model reasoning capabilities. Weaker models required multiple iterations to correct individual instances of the same error, while stronger models could identify and fix all instances simultaneously. As shown in the ablation study, removal of simulation feedback reduced a medium-capability model's performance down to that of simulation-assisted low-capability models. Simulation proved essential for preventing real-world failures, enabling the reliable detection of physical execution issues such as collisions, missing opening steps for robots with specific operational constraints, and invalid command sequences that would have caused failures in the physical laboratory. Importantly, the PCR experiments showed comparable performance to manual execution, demonstrating that automation through LLM-driven protocol generation does not compromise biological outcomes and can achieve the same scientific results as traditional manual approaches.

4.3 Current limitations and future work

Despite promising results, several limitations warrant attention in future work. LLMs still require substantial prompt engineering to achieve consistent and reliable performance, with sensitivity to prompt phrasing varying notably across model

families. Developing adaptive prompting strategies or domain-specific fine-tuning approaches tailored to laboratory protocols could reduce this dependence on manual prompt design. Current models also continue to struggle with reasoning about multi-well operations, complex volume calculations, and many-to-many transfer patterns—limiting their ability to generalize to more sophisticated assays. Addressing these challenges will be critical for extending automated protocol generation to increasingly complex experimental workflows.

The development of real-time liquid-handling monitoring represents another promising direction for future research. The current simulation validates protocols before execution by detecting physical motion errors, but cannot detect failures that occur during liquid handling at runtime, such as partial dispenses due to viscosity or air bubbles, missed dispenses, incorrect well targeting, or cross-contamination from tip reuse. In unsupervised operation, detecting such failures in real time and halting execution early may be preferable to completing a compromised experiment. Integrating sensor-based or vision-based liquid-handling monitors into the execution loop would extend PRISM's validation capabilities from pre-execution to runtime, complementing the existing simulation-based error detection.

The current Omniverse simulation has scope limitations, in particular regarding physical motion validation through reachability and collision feedback. Simulation fidelity constraints also exist, such as the lack of liquid physics modeling and the absence of chemical or biological process simulation, which require that scientific accuracy must be verified through other means. Expanding Omniverse error detection capabilities to include scientific accuracy validation and more detailed protocol adherence verification would strengthen the system's ability to catch errors before physical execution.

We also note that current text-based simulation feedback limits intuitive understanding of complex protocol execution flows for errors that are not expressed clearly in textual output formats. Recent work such as Agentic Lab[15] demonstrates the potential of vision-language models for laboratory reasoning, suggesting future directions for PRISM to incorporate multimodal simulation feedback—visualizing predicted equipment states, timing relationships, and potential failure modes alongside textual explanations to enhance protocol refinement. Frameworks such as ADePT[35] provide structured evaluation criteria for assessing such capabilities as they mature in autonomous laboratory settings.

Finally, we observe that simulation environments such as that considered here could be leveraged for laboratory design optimization, with digital twin modeling used to help identify laboratory configurations that maximize efficiency and minimize potential collision or access conflicts among robots.

4.4 Broader impact

The PRISM framework has the potential to democratize automated laboratory research by lowering the barriers to entry for institutions and researchers who lack specialized expertise in robotics and automation programming. By enabling scientists to describe experiments in structured natural language rather than requiring detailed knowledge of robot control systems, the PRISM approach makes laboratory automation accessible to a broader community. This democratization carries significant implications for reproducibility and standardization in biological protocols, as automatically generated protocols can be shared, validated, and executed consistently across different laboratories with compatible automation platforms.

The work presented here provides a pathway toward fully autonomous experimental design and execution, where the entire process from hypothesis formation through experimental execution and data analysis is automated. By bridging LLMs, physics-based simulation, and laboratory automation hardware, PRISM contributes to the foundation for self-driving laboratories. Such systems have the potential to accelerate scientific discovery while simultaneously reducing human error in protocol execution, enabling more reliable and reproducible experimental results. As these technologies mature and become more widely adopted, they may fundamentally transform how laboratory research is conducted, shifting the role of scientists from manual execution to higher-level experimental design and interpretation of results.

Author contributions

Conceptualization: R. S., A. R.; funding acquisition: R. S., I. F., A. R.; investigation: B. H., P. S., R. B., R. L., C. S., R. W., T. B.; methodology: B. H., P. S., R. B.; resources: R. S., A. R.; software: B. H., P. S., R. B.; writing – original draft: P. S., R. B.; writing – review & editing: P. S., R. B., I. F., A. R.

Conflicts of interest

There are no conflicts of interest to declare.

Data availability

The code and prompts for PRISM can be found at https://github.com/ramanathanlab/PRISM with DOI: https://doi.org/10.5281/zenodo.19797020. The version of the code employed for this study is v1.0.0 with DOI: https://doi.org/10.5281/zenodo.19797021. No primary datasets were generated; results reported in the manuscript are reproducible from the code in the archived repository.

Supplementary information (SI): full prompt templates for the protocol planning (stage 1) and protocol generation (stage 2) workflows. See DOI: https://doi.org/10.1039/d6dd00004e.

Acknowledgements

We acknowledge support from the U.S. Department of Energy, Office of Science, through the IDeA (Intelligent Design Assistant for Enzyme Discovery and Biosynthetic Pathway Optimization) project funded by the Office of Biological and Environmental Research, and the OPAL (Orchestrated Platform for Autonomous Laboratories to Accelerate AI-Driven BioDesign) project funded by the Office of Biological and Environmental Research and the Office of Advanced Scientific Computing Research, under Argonne National Laboratory contract DE-AC02-06CH11357.

Anthropic's Claude, Google's Gemini, and OpenAI's GPT models were used to assist in writing this manuscript's text.

References

1 S. Steiner, J. Wolf, S. Glatzel, A. Andreou, J. M. Granda, G. Keenan, T. Hinkley, G. Aragon-Camarasa, P. J. Kitson, D. Angelone and L. Cronin, *Science*, 2019, **363**, eaav2211.
2 Transcriptic/Strateos, *Autoprotocol*, https://autoprotocol.org, 2025, accessed: 2025-09-21.
3 J. Vrana, O. de Lange, Y. Yang, G. Newman, A. Saleem, A. Miller, C. Cordray, S. Halabiya, M. Parks, E. Lopez, S. Goldberg, B. Keller, D. Strickland and E. Klavins, *Synth. Biol.*, 2021, **6**, ysab006.
4 Y. Zhang, X. Chen, B. Jin, S. Wang, S. Ji, W. Wang and J. Han, *arXiv*, 2024, preprint, arXiv:2406.10833, DOI: 10.48550/arXiv.2406.10833.
5 M. C. Ramos, C. J. Collison and A. D. White, *Chem. Sci.*, 2025, **16**, 2514–2572.
6 S. Ren, P. Jian, Z. Ren, C. Leng, C. Xie and J. Zhang, *arXiv*, 2025, preprint, arXiv:2503.24047, DOI: 10.48550/arXiv.2503.24047.
7 A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White and P. Schwaller, *Nat. Mach. Intell.*, 2024, **6**, 525–535.
8 D. A. Boiko, R. MacKnight, B. Kline and G. Gomes, *Nature*, 2023, **624**, 570–578.
9 T. Song, M. Luo, X. Zhang, L. Chen, Y. Huang, J. Cao, Q. Zhu, D. Liu, B. Zhang, G. Zou, F. Zhang, W. Shang, J. Jiang and Y. Luo, *J. Am. Chem. Soc.*, 2025, **147**, 12534–12545.
10 Y. Ruan, C. Lu, N. Xu, Y. He, Y. Chen, J. Zhang, J. Xuan, J. Pan, Q. Fang, H. Gao, X. Shen, N. Ye, Q. Zhang and Y. Mo, *Nat. Commun.*, 2024, **15**, 10160.
11 O. O'Donoghue, A. Shtedritski, J. Ginger, R. Abboud, A. Ghareeb and S. Rodriques, *Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 2676–2694.
12 T. Inagaki, A. Kato, K. Takahashi, H. Ozaki and G. N. Kanda, *arXiv*, 2023, preprint, arXiv:2304.10267, DOI: 10.48550/arXiv.2304.10267.
13 U. B. Karli, J.-T. Chen, V. N. Antony and C.-M. Huang, *ACM/IEEE International Conference on Human-Robot Interaction*, 2024, pp. 361–370.
14 S. Jiang, D. Evans-Yamamoto, D. Bersenev, S. K. Palaniappan and A. Yachie-Kinoshita, *SLAS Technol.*, 2024, **29**, 100134.
15 W. Wang, S. Swain, J. Lee, Z. Lin, B. Canales, A. Aljović, Y. Liu, Q. Li, A. Marin-Llobet, M. Liu, Z. Gao, R. Liu, J. R. Alvarez-Dominguez and J. Liu, *bioRxiv*, 2025, preprint, DOI: 10.1101/2025.11.11.686354.
16 Y. Zhou, J. Yang, Y. Huang, K. Guo, Z. Emory, B. Ghosh, A. Bedar, S. Shekar, Z. Liang, P.-Y. Chen, T. Gao, W. Geyer, N. Moniz, N. V. Chawla and X. Zhang, *arXiv*, 2024, preprint, arXiv:2410.14182, DOI: 10.48550/arXiv.2410.14182.
17 J. A. Douthwaite, B. Lesage, M. Gleirscher, R. Calinescu, J. M. Aitken, R. Alexander and J. Law, *Frontiers in Robotics and AI*, 2021, **8**, 758099.
18 Y. Su, X. Wang, Y. Ye, Y. Xie, Y. Xu, Y. Jiang and C. Wang, *Chem. Sci.*, 2024, **15**, 12200–12233.
19 N. Yoshikawa, M. Skreta, K. Darvish, S. Arellano-Rubach, Z. Ji, L. Bjørn Kristensen, A. Z. Li, Y. Zhao, H. Xu, A. Kuramshin, A. Aspuru-Guzik, F. Shkurti and A. Garg, *Auton. Robots*, 2023, **47**, 1057–1086.
20 K. Darvish, M. Skreta, Y. Zhao, N. Yoshikawa, S. Som, M. Bogdanovic, Y. Cao, H. Hao, H. Xu, A. Aspuru-Guzik, A. Garg and F. Shkurt, *Matter*, 2025, **8**, 101897.
21 OpenAI, *GPT-5 System Card*, Large language model, 2025, https://cdn.openai.com/gpt-5-system-card.pdf.
22 Anthropic, *System Card Addendum: Claude Opus 4.1*, Large language model, 2025, https://www.anthropic.com/claude-opus-4-1-system-card.
23 Anthropic, *System Card: Claude Sonnet 4.5*, Large language model, 2025, https://www.anthropic.com/claude-sonnet-4-5-system-card.
24 Gemini Team, *Google, Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities*, Technical report, 2025, https://storage.googleapis.com/deepmind-media/gemini/gemini_v2_5_report.pdf.
25 Google DeepMind, *Gemini 2.5 Flash-Lite Model Card*, Large language model, 2025, https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-2-5-Flash-Lite-Model-Card.pdf.
26 R. Lewis, T. Ginsburg, D. Ozgulbas, C. Stone, A. Stroka, A. Cleary, I. Foster and N. Paulson, *J. Open Source Softw.*, 2026, **11**(119), 9416.
27 R. Vescovi, T. Ginsburg, K. Hippe, D. Ozgulbas, C. Stone, A. Stroka, R. Butler, B. Blaiszik, T. Brettin, K. Chard, M. Hereld, A. Ramanathan, R. Stevens, A. Vriza, J. Xu, Q. Zhang and I. Foster, *Digital Discovery*, 2023, **2**, 1980–1998.
28 NVIDIA, *Omniverse Platform for OpenUSD*, https://www.nvidia.com/en-us/omniverse/, 2024, accessed: 2025-10-14.
29 Opentrons Labworks Inc., *Opentrons OT-2 Automated Liquid Handling Robot*, https://opentrons.com/ot-2, 2024, accessed: 2025-10-14.
30 Precise Automation Inc., *PF400 Collaborative SCARA Robot*, https://preciseautomation.com/products/pf400/, 2024, accessed: 2025-10-14.
31 Hidex Oy, *Hidex Sense Microplate Reader*, https://www.hidex.com/instruments/hidex-sense/, 2024, accessed: 2025-10-14.
32 Azenta Life Sciences, *Automated Plate Sealer*, https://www.azenta.com/products/automated-plate-sealer, 2024, accessed: 2025-10-14.
33 Azenta Life Sciences, *Automated Plate Peeler*, https://www.azenta.com/products/automated-plate-peeler, 2024, accessed: 2025-10-14.
34 D. A. Boiko, R. MacKnight and G. Gomes, *arXiv*, 2023, preprint, arXiv:2304.05332, DOI: 10.48550/arXiv.2304.05332.
35 P. Salazar-Villacis and B. Benyahia, *Commun. Chem.*, 2026, **9**, 99.