

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

LLM-Guided Tool-Aware Task and Motion Planning for Chemistry Lab Automation

BOHYEONG PAK¹, HYUNHO CHO¹, and SANGHYUN KIM^{1,2}, (Member, IEEE)

¹Department of Mechanical Engineering (AgeTech-Service Convergence Major), Kyung Hee University, 1732 Deogyong-daero, Giheung-gu, Yongin-si 17104, Gyeonggi-do, Republic of Korea (e-mail: bhe1004@khu.ac.kr; chohh7391@khu.ac.kr)

²Advanced Institute of Convergence Technology, 145 Gwanggyo-ro, Yeongtong-gu, Suwon-si 16229, Gyeonggi-do, Republic of Korea

Corresponding author: Sanghyun Kim (e-mail: kim87@khu.ac.kr).

This research was supported by the SMEs Technology Innovation Development Program through the Technology Innovation and Promotion Agency (TIPA), funded by the Ministry of SMEs and Startups (RS-2024-00511332); the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (RS-2024-00411007); and a Korea Basic Science Institute (National Research Facilities and Equipment Center) grant funded by the Ministry of Science and ICT (RS-2025-00564593). Bohyeong Pak and Hyunho Cho contributed equally to this work.

ABSTRACT Autonomous chemistry laboratories can accelerate material discovery and improve experimental reproducibility. However, current systems are limited by rigid tool configurations and unreliable motion planning under orientation and collision constraints, and language-model-based planners stop at protocol generation without reasoning about execution context. This paper proposes a hierarchical, tool-aware task and motion planning framework for chemistry automation that couples large-language-model reasoning with batched differentiable optimization. The system converts natural-language instructions into validated chemical description language protocols, infers end-effectors and obstacle rearrangement strategies from procedural context, estimates the scene state through multi-view fiducial markers, and optimizes collision-free, constraint-satisfying trajectories in parallel, while a sensor-driven skill library handles tool changing and adaptive pouring. On a six-degree-of-freedom manipulator, the planner solves every configuration of three constrained tasks within the time budget, with each optimization round taking a narrowly bounded time per task, and on the task whose orientation constraint is most restrictive it reaches a 100.0% planning success rate against 73.3% for a sequential parameter-binding baseline. In randomized trials and ablation studies in an *NVIDIA Isaac Sim* testbed, the full framework executes complete chemical sequences involving tool switching, obstacle rearrangement, and multi-step reagent manipulation. On the physical robot, adaptive pouring reached a mean absolute error of 0.19–0.34 g for water and a syrup of about 3,000 cP and 0.36–0.73 g for powdered sugar over target masses of 10–50 g, and the Transfer procedure planned from the perceived positions was completed in all nine runs. The project page is available at https://rcilab.khu.ac.kr/auto_chem/.

INDEX TERMS Autonomous chemistry laboratory, intelligent planning, large language models, task and motion planning

I. INTRODUCTION

CHEMISTRY experiments are the cornerstone of material science discovery and hypothesis verification. Traditionally, these experiments have been conducted manually, a process that is resource-intensive and often limited by human variability in reproducibility. To address these limitations, Self-Driving Labs (SDLs) have emerged as a promising paradigm that automates the experimental cycle, from experiment selection and robotic execution to data acquisition [1], [2]. Despite this progress, existing laboratory automation pipelines are built around fixed tool configurations and protocol-specific routines, requiring manual re-

design whenever new apparatus or procedures are introduced. This rigidity does not scale as chemical workflows grow increasingly diverse and nonlinear, motivating a system that can autonomously adapt its tool-use configuration without manual domain redefinition.

Beyond operational flexibility, reliable physical execution remains difficult in laboratory environments. To address this challenge, constrained TAMP (Task and Motion Planning) [3], [4] has been applied to automate multi-step procedures such as pouring, stirring, and reagent transport. For example, Yoshikawa et al. [5] utilized PDDLStream [6] and projection-based sampling [7] to prevent reagent spillage under orienta-

tion constraints; however, their work reported that such task-specific constraints significantly degrade planning success rates on standard robot arms, requiring additional degrees of freedom to recover acceptable performance. This limitation is structural in approaches that independently sample each parameter and compose them through rejection: as the feasible region of the constraint manifold shrinks with decreasing kinematic redundancy, the probability that independently drawn samples jointly satisfy all constraints decreases. For 6-DoF manipulators commonly used in collaborative robot environments, this structural dependence results in degraded planning success rates under orientation constraints and unpredictable termination time, rendering such approaches unsuitable for time-sensitive chemical workflows.

To simultaneously address the challenges of context-aware tool selection and constrained planning reliability, this paper proposes the LLM-Guided Tool-Aware TAMP framework. A lightweight large language model (LLM) serves as a context-aware decision engine that extends beyond converting natural-language instructions into executable protocols. It determines the execution context required at each procedural stage by inferring the appropriate end-effector from object geometry and task requirements, and by deciding whether and where workspace obstacles should be rearranged through joint consideration of the current and subsequent procedures. These decisions are resolved before planning, allowing the TAMP solver to start from an updated configuration and focus on computing reliable, collision-free trajectories. To this end, the framework adopts cuTAMP [8], a differentiable TAMP solver that jointly optimizes batched candidate solutions, as its planning backbone, achieving both high planning success rates and structurally bounded planning time under tight orientation constraints on 6-DoF manipulators.

To support this execution model, a robust state grounding interface fuses multi-view *AprilTag*-based tracking with continuous sensor telemetry, providing the planning engine with consistent world-state estimates. Execution-level actions such as tool changing and adaptive pouring are handled by a set of sensor-feedback-driven skills that reliably bridge planned trajectories to physical execution. The main contributions of this paper are as follows:

- A hierarchical decision-making pipeline based on a lightweight LLM is established for context-aware chemistry automation. The pipeline converts natural language instructions into validated Chemical Description Language (XDL) protocols, infers the appropriate end-effector from object geometry and task requirements, and determines obstacle rearrangement strategies by jointly analyzing current and subsequent procedural context, enabling dynamic tool switching (e.g., suction, multi-finger grippers) without manual domain redefinition.
- cuTAMP is adopted as the planning backbone for constrained 6-DoF laboratory manipulation, jointly optimizing batched candidate solutions through differentiable gradient descent to produce constraint-satisfying trajec-

tories within a structurally bounded time budget. This simultaneously addresses the degradation in planning success under tight orientation constraints and the absence of any advance indication of when a plan will be returned, both of which make independent parameter sampling unsuitable for time-sensitive chemical workflows.

- These components are unified into a single end-to-end pipeline that enables fully autonomous execution of complex chemical workflows from a single natural-language command, demonstrating that operational flexibility and planning reliability can be addressed jointly rather than in isolation. The integrated framework is validated under randomized initial conditions in a high-fidelity simulation testbed built on *NVIDIA Isaac Sim*, covering module-level ablations and end-to-end performance on long-horizon workflows involving tool switching, obstacle rearrangement, and reagent handling. Adaptive pouring and the Transfer procedure are further validated on the physical FR5 with real reagents that differ in viscosity and form.

The remainder of this article is organized as follows: Section II reviews related work in autonomous chemistry, constrained TAMP, and LLM-based planning, and defines the position of this work. Section III details the proposed framework architecture and its core components. Section IV presents the experimental results obtained in the simulation testbed and on the physical FR5. Section V discusses the assumptions and limitations that bound the interpretation of these results. Finally, Section VI concludes the article.

II. RELATED WORK

A. AUTONOMOUS CHEMISTRY AND SELF-DRIVING LABORATORIES

Self-Driving Labs (SDLs) have emerged as a core technology for accelerating material discovery and ensuring experimental reproducibility. While early SDL research focused on automation tailored to specific experiments, the paradigm has recently shifted toward building flexible laboratories using general-purpose robots. Burger et al. [9] demonstrated autonomous photocatalyst optimization by introducing a mobile robot into the laboratory, and Fakhruldeen et al. [10] established a modular integration standard for heterogeneous robotic platforms through the ARChemist architecture. Lunt et al. [11] further extended this direction by integrating multiple robots to automate solid-state chemistry workflows.

Despite these advances, existing SDL systems share a common structural limitation: they are largely built around protocol-specific equipment and manually designed execution routines, requiring substantial reconfiguration when new procedures or apparatus are introduced. This rigidity extends to the manipulation layer as well. Kennedy et al. [12] and Huang et al. [13] developed sensor-feedback-based pouring skills, and Yoshikawa et al. [14] automated electrode polishing, yet these efforts assume fixed end-effectors tailored to predefined tasks, and incorporating a new tool

typically requires manual domain redefinition rather than autonomous adaptation. Recent efforts have also sought to improve the reliability of individual laboratory instruments through perception-based feedback. Khan et al. [15] integrated a YOLOv8-based computer vision model with the Opentrons OT-2 liquid handling robot to enable real-time detection of pipette tips and liquid volumes, demonstrating that closed-loop quality control can improve liquid handling accuracy. However, this approach remains scoped to a single, purpose-built instrument and a fixed operational domain, and does not address the broader challenge of adapting tool-use configurations across diverse procedural contexts. In summary, while automation of individual procedures has progressed considerably, adaptability remains limited when the required tools, vessel geometries, and procedural sequences are not known in advance.

B. TASK AND MOTION PLANNING FOR CONSTRAINED LABORATORY MANIPULATION

Complex chemistry experiments simultaneously require long-horizon task planning and precise motion planning under strict physical constraints. PDDLStream, proposed by Garrett et al. [6], has become a representative framework for combining symbolic logic with sampling-based planning. Existing studies have applied PDDLStream to enforce geometric constraints such as orientation maintenance during liquid transfer; however, PDDLStream operates by first constructing an optimistic plan and then sequentially evaluating the stream instances that support it, attempting to bind each continuous parameter independently without accounting for its interaction with other constraints. As a result, as kinematic redundancy decreases and the feasible region satisfying all constraints simultaneously narrows, the probability of finding a valid binding decreases structurally. Furthermore, because the number of sampling iterations required to find a valid solution is inherently stochastic, the termination time of such planners is not known in advance, which is a critical limitation for chemistry laboratory automation, where unexpected planning delays can disrupt time-sensitive reaction conditions.

Optimization-based motion planning by Muchacho et al. [16] and constraint-satisfying sampling techniques by Kingston et al. [17] have been investigated to improve constraint handling, yet they do not fundamentally resolve the reliability degradation on manipulators with limited degrees of freedom, nor do they guarantee bounded planning latency. Coleman et al. [18] proposed a design to lower the barrier to entry for complex robotic software through a *MoveIt!* case study, but the underlying challenges of planning reliability and latency predictability remain unaddressed. Hybrid approaches combining optimization-based pre-grasp planning with reinforcement learning have also been explored for dynamic environments. Li et al. [19] proposed a method that integrates quasi-Newton trajectory initialization with a recurrent deep RL policy for dynamic obstacle avoidance and grasping in mobile manipulators. However, learning-based planners of this kind do not provide formal guarantees on

constraint satisfaction or planning latency, as convergence depends on stochastic policy training rather than deterministic optimization over a fixed computational budget. This limitation extends to the broader class of end-to-end learned policies: recent vision-language-action (VLA) models for manipulation [20] and their counterparts for driving that fuse navigation context into the action model [21] directly map visual observations and language instructions to actions and generalize flexibly across diverse tasks, yet they likewise offer no formal guarantees on constraint satisfaction or bounded planning latency, both of which are indispensable for time-sensitive chemical workflows. Regardless of the underlying approach, these methods have primarily been evaluated on general pick-and-place scenarios. Laboratory manipulation poses a compounded difficulty: orientation constraints must be maintained throughout liquid transport, obstacles may need to be rearranged between procedural steps depending on task context, and tool switching must be coordinated within multi-step workflows, all on low-redundancy robots where planning reliability is already degraded. This combination has received limited attention in existing TAMP literature.

C. LARGE LANGUAGE MODELS FOR ROBOT PLANNING AND EMBODIED DECISION-MAKING

The advent of Large Language Models (LLMs) has opened new possibilities for translating abstract natural language commands into robot-executable instructions. Mehr et al. [22] proposed the Chemical Description Language (XDL) standard for the digitization of chemical literature, and Yoshikawa et al. [23] built the CLAIRify system, which uses LLMs to convert chemical texts into robot commands. Darvish et al. (ORGANA) [24] presented an interactive robotic assistant that utilizes an LLM-based planner to converse with scientists and assist with experiments, and ChemCrow [25] and Coscientist [26] demonstrated that LLMs can autonomously search for chemical tools and design experimental procedures.

While these systems demonstrate strong capabilities in semantic decomposition and high-level task sequencing, their use of LLMs generally remains confined to text-based procedure generation or human decision-making assistance. The generated plans are typically handed off to a separate, manually configured execution pipeline, and the LLM itself is not required to reason about the physical execution environment, such as which end-effector is geometrically appropriate for a given vessel, or whether workspace obstacles must be rearranged before a procedure can proceed. As a consequence, geometric validity, collision avoidance, and orientation feasibility must be delegated entirely to downstream planning modules. This pattern is exemplified by general-purpose LLM-robotics interfaces such as that of Vemprala et al. [27], in which an LLM composes robot behaviors across manipulation and navigation tasks by invoking a predefined high-level function library, so that geometric feasibility and physical execution are resolved by that manually designed layer rather than reasoned about by the language model itself. Recent work has sought to close this gap through simulation-

based feasibility verification. Qu et al. [28] proposed TRTP, a three-stage framework that couples vision-language models with digital twin simulation to evaluate spatial reachability and physical feasibility of LLM-generated action sequences before execution. While this closed-loop architecture improves plan reliability, the feasibility check is performed in a separate simulation module rather than being resolved at the symbolic reasoning stage; end-effector selection and context-dependent obstacle rearrangement remain outside the scope of the language model's decision-making. This separation means that existing LLM-based robot planning systems can formulate what to do, but lack the ability to determine the execution context required to carry it out, namely which tool is appropriate for the current task and whether the workspace must be reconfigured before execution proceeds.

D. POSITION OF THIS WORK

Existing work has addressed individual aspects of autonomous chemistry from complementary directions. SDL and chemistry automation systems have demonstrated automated experimental execution, but remain largely protocol-specific and assume fixed tool configurations that require manual redesign for new procedures. Constrained TAMP approaches have tackled motion planning under geometric constraints, but exhibit reliability degradation on low-redundancy manipulators and do not guarantee bounded and predictable planning latency for time-sensitive chemical workflows. LLM-based robot planning systems have enabled high-level semantic reasoning for chemical procedures, but remain confined to protocol generation without determining the execution context required to carry out each procedure.

The proposed framework is positioned at the intersection of these three directions, unifying LLM-based context-aware decision-making with batched differentiable optimization-based planning within a single end-to-end system.

III. METHODS

The framework addresses the three limitations identified in Section II-D with four modules. The LLM Module addresses the first and third by converting natural-language instructions into validated XDL protocols and determining the execution context of each procedure, namely end-effector selection and obstacle rearrangement. The TAMP Module addresses the second by adopting cuTAMP [8] as its planning backbone. The Skill Library Module handles through sensor feedback the execution-level actions that geometric planning alone cannot, such as adaptive pouring and temperature regulation. The Perception Module underpins all three by constructing the World State, a structured representation of object poses, environment geometry, and apparatus configurations.

A. FRAMEWORK OVERVIEW

Fig. 1 illustrates the overall architecture and inter-module data flow of the proposed framework, in which the LLM Module, TAMP Module, and Skill Library Module operate sequentially on the basis of a shared World State continuously

maintained by the Perception Module, which senses object poses and sensor data throughout the laboratory environment.

The pipeline begins with the LLM Module converting the natural-language command into an XDL protocol. Since a linguistically valid protocol may still be physically infeasible in the current workspace, the protocol is cross-checked against the current World State before being passed to the execution stage. Prior to each XDL procedure, the LLM Module infers the required end-effector, whether obstacle rearrangement is necessary, the auxiliary end-effector for rearrangement, and the target location for the relocated object by referencing the latest World State.

The TAMP Module receives these symbolic decisions, constructs a task plan, and computes a collision-free, constraint-satisfying nominal trajectory for each operator by searching for a plan skeleton and using the latest World State. However, since geometric planning alone cannot account for physical uncertainties that arise during execution, such as liquid flow and contact dynamics, the nominal trajectories are passed to the Skill Library Module. The Skill Library Module translates the nominal trajectories into physical execution through sensor-feedback-based motion primitives, correcting for uncertainties not modeled at the planning stage in real time.

B. LLM MODULE

The LLM Module is the symbolic reasoning layer of the framework. It translates natural-language instructions into executable protocols and resolves context-dependent execution decisions before physical planning. Its design confines the LLM to discrete symbolic outputs, because continuous 3D spatial reasoning, such as deriving exact placement coordinates or collision-free waypoints, lies outside the reliable operating regime of current language models. The module therefore selects an end-effector from a predefined inventory, decides whether obstacle rearrangement is necessary, and specifies a target workspace grid rather than a Cartesian location, while the TAMP Module resolves the exact placement in continuous space. This division keeps hallucinated spatial quantities out of the planning pipeline.

The module proceeds through three stages. The XDL Generator synthesizes a protocol from the command, the XDL Validator checks it in three progressively stricter stages, and the Action Reasoner infers tool selection and obstacle rearrangement at each procedural step.

1) Global Experimental Protocol Synthesis

The XDL Generator functions as a schema-constrained protocol generation module that maps natural-language commands into complete, end-to-end structured XDL programs encoding the full experimental procedure in a single pass. Because the output space is confined to a fixed XDL schema with a finite set of operators and parameter types, a lightweight *Llama 3.2 1B* model fine-tuned via Low-Rank Adaptation (LoRA)-based supervised fine-tuning (SFT) is sufficient to achieve reliable conversion without the overhead and latency of general-purpose large models. The training dataset con-

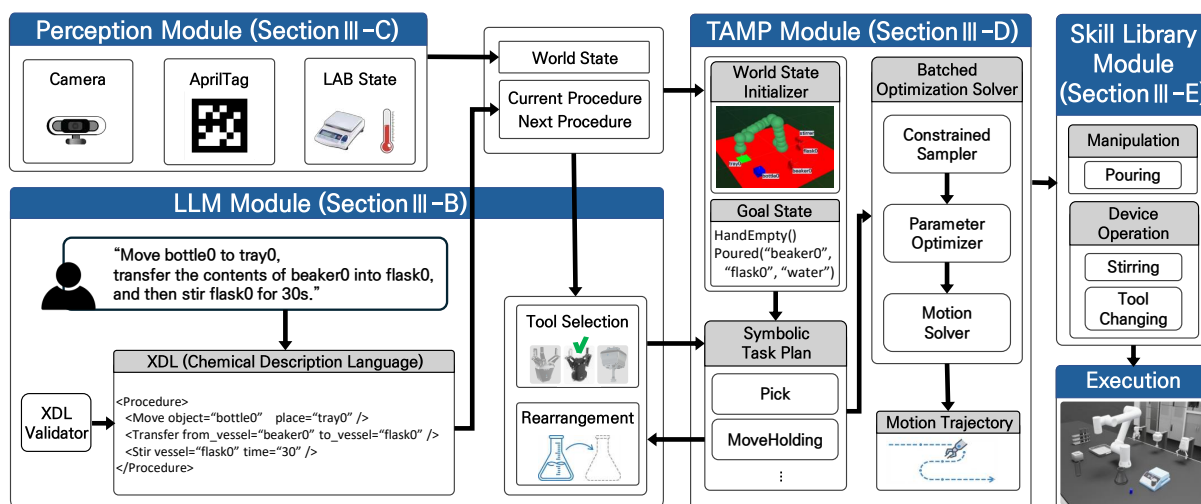


FIGURE 1. Architectural overview of the LLM-Guided Tool-Aware TAMP framework. The hierarchy illustrates the integration of symbolic reasoning (LLM Module) with physical execution (TAMP Module) supported by a multi-modal perception interface (Perception Module) and a robotic skill library (Skill Library Module).

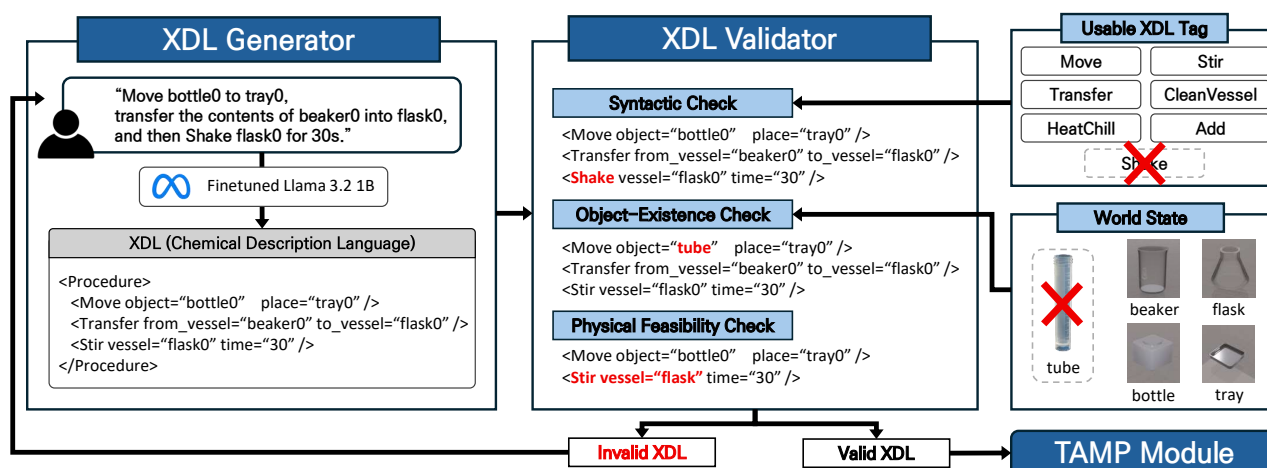


FIGURE 2. The XDL Generator and Validator pipeline within the LLM Module. A fine-tuned *Llama 3.2 1B* model converts a natural language command into a structured XDL protocol, which is then subjected to three progressively stricter checks (syntactic, object-existence, and physical-feasibility) before a valid protocol is forwarded to the TAMP Module.

sists of 5,000 combinatorially constructed samples covering synonym variations for six core XDL operators, continuous numerical parameter ranges, and diverse object instances. Additional details on dataset construction and training are provided in Section IV-A3.

2) Protocol Validation

The XDL Validator serves as the safety interface between the language layer and the physical execution pipeline. While the XDL Generator produces structured protocols, a protocol that is syntactically valid may still be physically infeasible. The Validator therefore enforces three progressively stricter checks before any robot motion is executed. Syntactic validation verifies well-formed XML structure and permitted operator tags, blocking case-sensitivity mismatches and undefined elements. Object-existence validation cross-references all referenced objects against the workspace inventory provided by the Perception Module, intercepting commands that

refer to objects absent from the current environment, and also checks that the quantities and objects of the protocol appear in the natural-language command. Physical-feasibility validation simulates the content state of each vessel across the procedure and rejects sequences that violate procedural preconditions, such as stirring an empty vessel or transferring from an empty source. If a conflict is detected at any stage, execution is halted and the offending step and error type are reported to the user, enabling targeted protocol-level revision before physical failure can occur. The conditions the three checks verify are enumerated as 25 items in Table 11, ten for syntactic validation, six for object existence, and nine for physical feasibility. Conditions outside this enumeration are not resolved here. Whether two reagents may be combined beyond the enumerated pairs is not a judgement this system makes, and physical constraints such as reachability are resolved by the TAMP Module. Fig. 2 illustrates representative examples: syntactic validation rejects undefined operator tags

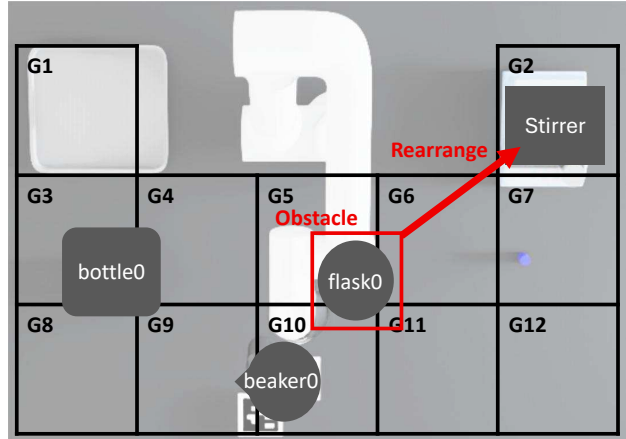
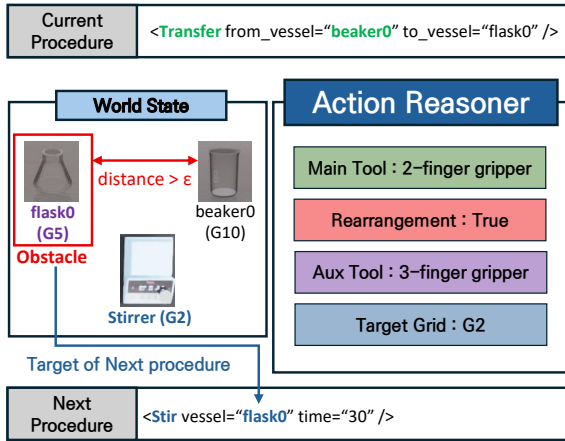


FIGURE 3. Illustration of the Action Reasoner pipeline. Given the current and next XDL procedures alongside the World State, a fine-tuned *Llama 3.2 1B* model infers four discrete outputs: the main end-effector, whether obstacle rearrangement is necessary, the auxiliary tool for rearrangement, and the target workspace grid within the discretized workspace (G1–G12).

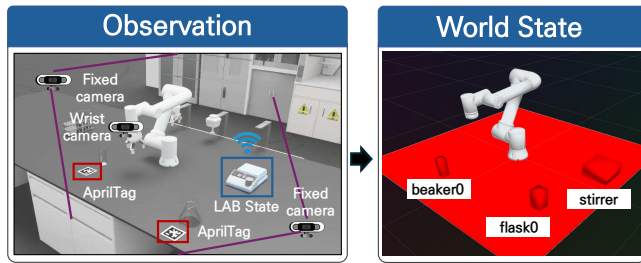


FIGURE 4. Overview of the Perception Module pipeline. Multi-view observations of the laboratory environment are fused and transformed into a World State, providing downstream modules with consistent 6-DoF pose estimates of laboratory assets.

such as Shake, object-existence validation intercepts references to absent objects such as tube, and physical-feasibility validation rejects operations such as stirring an empty vessel.

3) Context-Aware Tool and Action Reasoning

The Action Reasoner determines the end-effector and obstacle rearrangement strategy for each procedure before the TAMP Module begins planning. This stage is critical for reducing planning failures caused by tool mismatches or obstructed workspaces. Following the symbolic-level design principle established in Section III-B, the module operates on the discretized workspace representation illustrated in Fig. 3, where the environment is partitioned into 12 workspace grids (G1–G12) based on the spatial state provided by the Perception Module. A fine-tuned *Llama 3.2 1B* model receives four inputs: the current XDL procedure, the subsequent procedure, the current obstacle layout, and the workspace grid state. From these, it infers four discrete decision variables: the required end-effector for the current operation, whether object rearrangement is necessary, the auxiliary tool for rearrangement, and the target grid for the relocated object.

This design provides two main advantages. First, it enables semantic tool assignment by grounding the model’s knowledge of object affordances in the physical constraints of the

available end-effectors. Consequently, appropriate grippers can be assigned even for vessel types unseen during training, minimizing the need for manual domain redefinition. Second, it enables context-aware rearrangement by jointly considering the current and subsequent procedures, allowing obstacles to be moved proactively toward regions required for future steps and thereby reducing redundant rearrangements. The context the module resolves is of a bounded kind: it identifies which objects the next procedure acts on and chooses where to put an obstacle from its relation to those objects. The wider purpose of an experiment and the chemical soundness of a sequence are outside what it reasons about. For example, as illustrated in Fig. 3, when the flask obstructs access to the beaker in the current procedure, and the next procedure is a Stir operation targeting the flask, the Action Reasoner identifies the flask as an obstacle requiring rearrangement and infers G2 as the target grid, since the stirrer is located there.

C. PERCEPTION MODULE

The Perception Module grounds the World State. It provides the TAMP Module with the 6-DoF poses of laboratory assets, the Action Reasoner with the workspace grid state, and the Skill Library Module with continuous sensor streams.

The module adopts an *AprilTag*-based strategy [29] to achieve robust pose estimation at low computational latency, as illustrated in Fig. 4. Markers are rigidly affixed to each vessel, and their poses are estimated by three cameras. Two calibrated RGB cameras mounted at spatially separated positions around the workspace estimate the poses continuously, and a camera mounted on the wrist supplements them. Fusing observations from multiple viewpoints in this manner follows established multi-view planar-marker tracking approaches [30] that improve pose consistency and robustness to occlusion relative to single-camera estimation.

1) Lab Object Localization

Each lab object is assigned a unique identifier, and its localization proceeds through a three-stage pipeline: pose com-

putation in the robot base frame, dual-camera fusion against occlusion, and tag-to-vessel coordinate transformation.

Single-Camera Pose Computation: For each camera $c \in \{1, 2\}$, the pose of object i in the robot base frame is obtained by chaining the fixed extrinsic calibration bT_c with the detected marker pose ${}^cT_t^{(i)}$. To assess observation reliability, the Euclidean distance $d_c^{(i)}$ from the camera optical center to the marker is recorded as a weight metric:

$${}^bT_{t,c}^{(i)} = {}^bT_c \cdot {}^cT_t^{(i)}, \quad d_c^{(i)} = \|t({}^cT_t^{(i)})\|_2. \quad (1)$$

Inverse-Distance Weighted Fusion: When both cameras yield valid observations, the estimates are fused by assigning higher confidence to the camera closer to the marker, thereby minimizing distance-dependent projection errors. Given a regularization constant $\varepsilon = 10^{-6}$ to prevent division by zero, the normalized weights W_c and the resulting fused translation $\tilde{t}$ and orientation $\tilde{q}$ via Spherical Linear Interpolation (SLERP) [31] are computed as follows:

$$W_c = \frac{(d_c^{(i)})^{-2}}{\sum_{j=1}^2 (d_j^{(i)})^{-2} + \varepsilon}, \quad (2)$$

$$\tilde{t} = W_1 t_1 + W_2 t_2, \quad \tilde{q} = \text{SLERP}(q_1, q_2, W_2).$$

If occlusion occurs and only one camera provides a valid observation, its estimate is used directly without fusion to maintain continuity of pose tracking.

Tag-to-Object Offset Transform: Since markers are affixed to the side surfaces of vessels rather than their geometric centers, a fixed rigid offset tT_o is applied to recover the true physical center of the object:

$${}^bT_o^{(i)} = {}^b\tilde{T}_t^{(i)} \cdot {}^tT_o. \quad (3)$$

The resulting transform ${}^bT_o^{(i)}$ is broadcast over the ROS 2 TF tree at 30 Hz. These localized poses form the raw spatial foundation from which the World State is assembled, as described in Section III-C2. When the two fixed cameras cannot read a marker or their measurement cannot be trusted, the arm carries the wrist camera through predefined viewpoints to search for the marker. If the wrist camera cannot read the marker either, execution stops and the object is reported to the user, as when XDL validation fails.

2) State Grounding and World Modeling

The 6-DoF poses broadcast by the Perception Module are assembled into the World State, a unified environmental representation that the TAMP Module consumes for motion planning. The World State is initialized at the beginning of each XDL procedure and updated after every stage transition, ensuring that the planner always operates on the current laboratory configuration. To populate the World State, the module partitions all objects in the workspace into two categories: target objects designated for manipulation in the current stage, and non-target objects treated as obstacles. Static elements such as tables and laboratory equipment are maintained as fixed obstacles, while movable objects such as beakers, bottles, and flasks are dynamically refreshed from

the latest Perception Module output at each update. To keep planning tractable as the number of laboratory objects grows, all object models are represented using simplified geometric primitives rather than full CAD geometries. Target objects are approximated as axis-aligned boxes or cylinders, and collision is evaluated by covering these primitives with spheres, which provides a conservative margin at the faces and edges. This abstraction reduces the cost of collision checking and path sampling while preserving execution feasibility, so the TAMP Module scales to cluttered workspaces.

3) Multi-Modal Sensor Stream Processing

Beyond geometric world-state grounding, the Perception Module acts as a centralized telemetry hub that aggregates, filters, and distributes continuous data streams from physical sensors. This real-time sensor processing is critical for closing the control loop in the Skill Library Module during physical execution, such as in adaptive pouring or temperature regulation. The module processes two sensory modalities:

Scale Data: For precise quantitative liquid transfer, the module interfaces with an electronic precision scale. The raw weight measurements are acquired, synchronized with the system clock, and processed into a stable data stream. This filtered scale data is directly consumed by the Adaptive Pouring skill, which uses it to estimate the flow rate and adjust the tilt of the vessel (Section III-E1).

Temperature Data: To ensure strict adherence to chemical reaction conditions, the module also integrates temperature readings from external laboratory equipment, such as hotplate thermocouples. Raw temperature readings are monitored and smoothed to reject transient electrical noise. This provides a stable thermal state stream, which the device operation skills use to verify reaction temperatures and, if necessary, suspend execution until the target conditions are met.

By consolidating and filtering these heterogeneous data streams, the Perception Module shields the high-level planning and execution modules from hardware-specific noise, providing a clean and synchronized sensory interface for complex chemical manipulation.

D. TASK AND MOTION PLANNING MODULE

The TAMP Module computes collision-free, constraint-satisfying trajectories for multi-step manipulation procedures in chemistry experiments while keeping planning latency structurally bounded and predictable, as required by time-sensitive chemical workflows. As reviewed in Section II-B, the PDDLStream-based approach [6], which attempts to bind each continuous parameter independently, is structurally ill-suited to these requirements. To address this limitation, the TAMP Module is designed as an integrated planning framework that combines cuTAMP [8] with chemistry-specific operators and an LLM-guided tool-switching mechanism.

1) Chemistry-Specific TAMP

The TAMP Module adopts cuTAMP as its planning solver, formulating the planning problem as a parallelized differen-

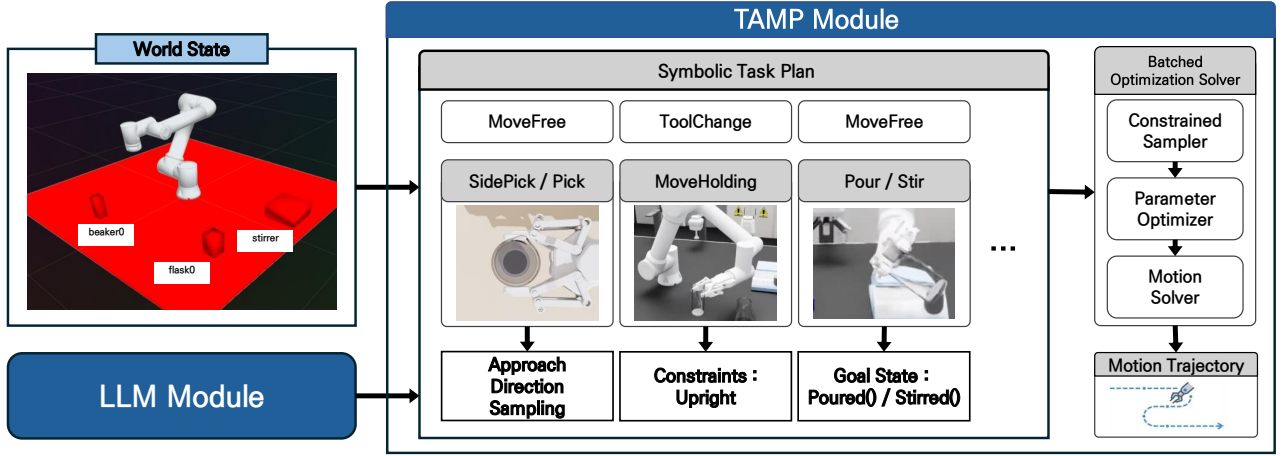


FIGURE 5. Overview of the TAMP Module. The World State and the symbolic task plan, comprising chemistry-specific operators (SidePick, MoveHolding, Pour, Stir) alongside base operators (MoveFree, Pick), are passed to the Batched Optimization Solver, which sequentially executes a Constrained Sampler, Parameter Optimizer, and Motion Solver to produce the final motion trajectory.

tiable optimization process that simultaneously drives a large batch of candidate particles toward the constraint manifold via gradient-based optimization. Unlike approaches that attempt to bind each parameter independently, cuTAMP jointly optimizes all constraints across batched candidate solutions, enabling convergence toward the feasible region even under tight orientation constraints on low-redundancy manipulators. Because a fixed number of optimization steps is executed over a fixed batch of particles, the time of one optimization round falls within a narrow range, and a round that finds no solution is restarted with fresh particles. Planning latency can therefore be estimated in units of rounds, so that planning can be scheduled around time-sensitive reaction conditions such as temperature control and reagent timing.

Chemistry laboratory manipulation imposes requirements beyond those of general pick-and-place tasks. Cylindrical vessels must be grasped stably despite their geometry, liquids must be transported without spillage and transferred in precise quantities, and device-level operations such as stirring must be coordinated within the planning pipeline. To meet these requirements, we introduce four chemistry-specific operators in addition to the base TAMP operators: SidePick, for grasping the lateral surface of cylindrical vessels such as beakers and flasks; MoveHolding, for maintaining an upright orientation throughout the trajectory during liquid transport; Pour, for planning the tilting motion after the robot reaches the top of the target vessel; and Stir, a device-level operator for controlling the magnetic stirrer. Fig. 5 illustrates the overall structure of the TAMP Module, showing how chemistry-specific operators are composed into a symbolic task plan and resolved by the batched optimization pipeline.

The purpose of SidePick is to enable stable grasping of cylindrical vessels from any approach direction, where top-down grasping would otherwise be insufficient. In the object coordinate frame, where the z-axis is aligned with the vertical axis of the cylindrical vessel and the origin is located at the geometric center of the object, a grasp pose $\mathbf{g} = (\mathbf{t}, \mathbf{r})$ consists of translation $\mathbf{t} \in \mathbb{R}^3$ and orientation $\mathbf{r} = (r, p, y)$. For full-

circumference grasping of cylindrical objects, the approach angle α is sampled uniformly in the horizontal plane:

$$\alpha \sim \mathcal{U}(0, 2\pi). \quad (4)$$

The grasp orientation is determined as follows:

$$r = 0, \quad p = \alpha, \quad y \in \left\{ -\frac{\pi}{2}, -\frac{\pi}{2.5}, \frac{\pi}{2.5}, \frac{\pi}{2} \right\}, \quad (5)$$

where roll r is fixed to zero such that the gripper x-axis aligns with the vertical direction, pitch p determines the approach direction around the object circumference via the approach angle α so that the gripper z-axis (finger approach direction) points horizontally toward the object center, and yaw y is discretely sampled from gripper tilt candidates. Discrete yaw sampling diversifies the gripper contact configuration on the vessel surface, thereby improving grasp stability across varying approach conditions, as illustrated in Fig. 12. The grasp height along the object z-axis is sampled within a range centered at the object origin:

$$t_z \sim \mathcal{U}\left(-\frac{h_o}{3}, \frac{h_o}{3}\right), \quad (6)$$

where the lateral position in the horizontal plane is determined by the approach direction α at a fixed radial offset from the object surface. This full-circumference sampling strategy improves search efficiency during particle initialization and enables stable grasping of rotationally symmetric vessels from any direction.

MoveHolding is an operator that enforces an upright constraint throughout the entire trajectory while the robot moves with a grasped object. It is introduced to prevent liquid spillage caused by vessel tilting during transport. The opening direction vector $\mathbf{u}_o$ of the grasped object is derived from the robot joint variables $\mathbf{q}$ and the fixed grasp transform $\mathbf{R}_{\text{grasp}}$ determined at the time of grasping:

$$\mathbf{u}_o(\mathbf{q}) = \mathbf{R}_{ee}(\mathbf{q}) \cdot \mathbf{R}_{\text{grasp}}^{-1} \cdot \mathbf{z}_w, \quad (7)$$

where $\mathbf{R}_{ee}(\mathbf{q})$ is the end-effector rotation matrix obtained from forward kinematics and $\mathbf{z}_w = [0, 0, 1]^T$. Since $\mathbf{R}_{\text{grasp}}^{-1} \cdot \mathbf{z}_w$

Algorithm 1 LLM-Guided Adaptive TAMP with Tool Switching

Require: XDL Procedure X_i , Next Procedure X_{i+1} , World State S , Current Tool T

Ensure: Motion Trajectory τ

```

1:  $[T_m, b_r, T_a, G^*] \leftarrow \text{LLM\_Reasoning}(X_i, X_{i+1}, S)$ 
2: if  $b_r$  is true then
  {Rearrangement required}
3:   if  $T \neq T_a$  then
4:      $\text{Execute\_Tool\_Change}(T \rightarrow T_a)$ 
5:      $T \leftarrow T_a$ 
6:   end if
7:    $S \leftarrow \text{Update\_World\_State}()$  {Ground perception before rearrangement}
8:    $\tau_a \leftarrow \text{cuTAMP\_Planner}(\text{Pick-MoveHolding-Place}, S, G^*)$ 
9:    $\text{Execute}(\tau_a)$ 
10: end if
11: if  $T \neq T_m$  then
  {Tool change for main task if needed}
12:    $\text{Execute\_Tool\_Change}(T \rightarrow T_m)$ 
13:    $T \leftarrow T_m$ 
14: end if
15:  $S \leftarrow \text{Update\_World\_State}()$  {Final state update for main task}
16:  $\tau_m \leftarrow \text{cuTAMP\_Planner}(X_i, S, T_m)$ 
17: return  $\tau_m$ 

```

is a constant vector computed once at the time of grasping, this constraint can be evaluated solely from the robot joint variables without requiring real-time object pose estimation. The alignment error between this vector and the world z-axis is formulated as an upright cost J_{up} , which is zero when the object is vertical and increases as it tilts:

$$J_{up}(\mathbf{q}) = 1 - \mathbf{u}_o(\mathbf{q}) \cdot \mathbf{z}_w. \quad (8)$$

When $J_{up} = 0$, the object is perfectly upright. For a maximum tilt angle $\theta_{\max}$, the constraint is satisfied when

$$J_{up}(\mathbf{q}) \leq 1 - \cos(\theta_{\max}), \quad (9)$$

and the same constraint is also applied to the Place operator.¹

Pour is an operator responsible for the tilting motion after the robot has reached the top of the target vessel. It is designed to decouple the transport phase from the actual pouring action, allowing the solver to focus on geometric validity and collision avoidance while delegating precise quantitative transfer to the Skill Library Module. The TAMP Module generates a geometric trajectory that tilts the opening of the vessel toward the target vessel, ensuring a valid starting pose and collision avoidance for the pouring action. This trajectory serves as the initial condition for the Adaptive Pouring skill described in Section III-E1, which estimates the flow rate from real-time weight feedback from an electronic scale and adjusts the tilt of the vessel until the target amount is reached. Upon completion of the Pour operator, the state of the corresponding vessel transitions to Poured.

¹The admissible tilt follows from the vessel geometry and the fill depth. The beakers used in the physical evaluation have a radius of 30 mm and a height of 60 mm and were filled to a depth of 50 mm. Requiring the liquid surface to stay at least 5 mm below the rim admits a tilt of 9.5°. We set $\theta_{\max}$ to 5°, below that value, to leave a margin for the liquid surface sloshing under acceleration and deceleration during transport. Volume conservation makes this bound independent of fluid viscosity.

Stir is a device-level operator that transmits the target RPM and stirring duration to the magnetic stirrer. By directly invoking the device operation skill in the Skill Library Module without involving motion planning, it reflects the design principle that device-level operations are handled entirely at the execution layer. The stirrer operates at the specified RPM for the set duration, and upon completion of the Stir operator, the state of the corresponding vessel transitions to Stirred.

If task planning or motion planning fails during any XDL procedure, the procedure is retried while the time budget remains. Upon a task-planning retry, the plan skeleton search is re-executed with new initial conditions; upon a motion-planning retry, particles are re-initialized and optimization is performed again. If a valid solution is not found before the budget is spent, the system halts the workflow and returns an error message containing the failure cause and the relevant procedure information, allowing the user to diagnose and revise the experimental protocol.

2) LLM-Guided Adaptive Tool Switching and Auxiliary Action Insertion

Given the decisions made by the Action Reasoner, including which tool to mount, whether rearrangement is needed, and where to place the obstacle, this section describes how these discrete outputs are translated into executable steps within the TAMP pipeline. Tool switching is handled as a pre-execution step decoupled from the symbolic domain, while auxiliary rearrangement is inserted as a preceding Pick-MoveHolding-Place sequence when required. The full procedure is summarized in Algorithm 1. Here, T_m denotes the required tool for the main task, T_a the auxiliary tool for rearrangement, b_r a boolean flag indicating whether rearrangement is necessary, and G^* the target grid for the relocated object.

Tool changes are treated as pre-execution steps independent of the TAMP symbolic domain, thereby avoiding domain-redefinition overhead whenever the hardware configuration changes. When required, cuTAMP generates a collision-free trajectory to the tool exchange point, after which the physical exchange is completed by the Tool Changing skill in the Skill Library Module.

When rearrangement is required, an auxiliary Pick-MoveHolding-Place sequence is inserted before the main procedure, with a tool change to the auxiliary tool performed first if necessary. The target zone is selected to reflect the context of the subsequent procedure, thereby reducing unnecessary rearrangement in later steps and improving overall execution efficiency. The stage-wise world-state update before and after each procedure ensures that plans reflect all preceding changes, including object positions, tool configuration, and auxiliary rearrangement outcomes.

E. SKILL LIBRARY MODULE AND EXECUTION

The Skill Library Module is an execution-layer module that instantiates the nominal trajectories generated by the TAMP Module into sensor-feedback-based motion primitives at execution time. Since TAMP considers only geometric con-

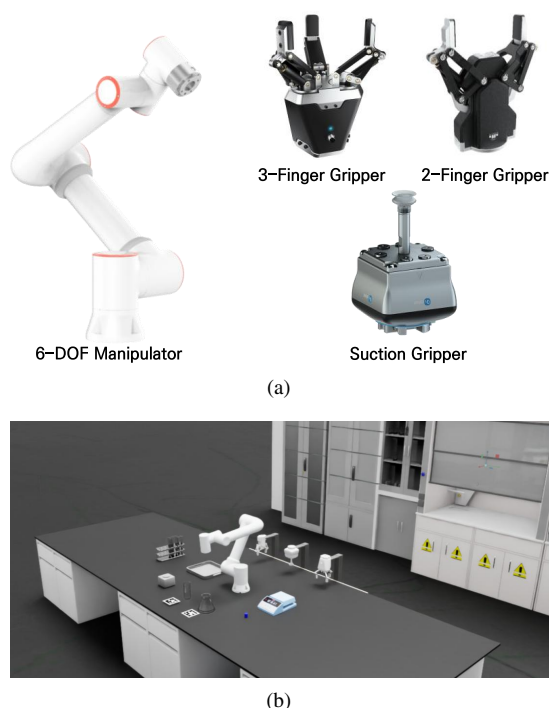


FIGURE 6. Hardware components and simulation environment for experimental evaluation. (a) The 6-DoF collaborative manipulator (FAIRINO-FR5) alongside three specialized end-effectors: the 2-finger gripper, 3-finger gripper, and suction gripper. (b) The simulation testbed is constructed using NVIDIA Isaac Sim.

straints and logical sequencing, the Skill Library invokes the appropriate skill mapped to each TAMP operator to handle physical variables not modeled at the planning stage. The module is organized into two categories: manipulation skills and device operation skills.

1) Manipulation Skills

This category comprises two skills: Adaptive Pouring and Automatic Tool Changing. The Adaptive Pouring skill corrects the nominal trajectory of the Pour operator using real-time weight feedback provided by the Perception Module (Section III-C3). Because the flow rate of a reagent is difficult to predict, the nominal trajectory alone cannot achieve precise quantitative transfer. The scale measures only the accumulated weight of the received reagent, so the skill fits a line to the weight readings over a recent window and uses its slope as the flow rate. On reaching the pouring pose, the vessel is tilted slowly until the reagent is first detected on the scale, and the tilt one scale measurement period (0.2 s) earlier is taken as the flow-onset angle. Surface tension holds the liquid at the lip until a certain angle is exceeded, and that angle depends on the vessel, the reagent, and the amount remaining, so the flow-onset angle is not fixed in advance but detected in this way on every pour. The relation between tilt and flow rate is then estimated continuously while pouring, and the tilt is adjusted to follow a target flow rate proportional to the amount remaining, so the flow slows as the target approaches. The amount that has not yet reached the scale and the amount

that will flow out while the vessel is righted are predicted from this relation, and the vessel is returned below the flow-onset angle once the measured weight plus the predicted amount reaches the target. If the weight is still short of the target after it settles, a correction pulse is repeated. The vessel is tilted slowly until flow is detected again, held for the time that releases slightly less than the remaining difference, and returned. Powdered reagents use the same structure with a small periodic oscillation added to the tilt to induce flow.

The Automatic Tool Changing skill completes the tool exchange as a concrete hardware operation: since the TAMP Module plans the exchange solely as a geometric motion to the tool rack, control signals are applied to the electromagnet upon reaching the exchange point and the mechanical latching sequence is executed to complete the physical coupling.

2) Device Operation Skills

Device operation skills translate device commands, which TAMP treats as abstract symbols, into actual communication protocols and sensor feedback loops; unlike the immediate state transitions assumed by TAMP, physical laboratory devices require time to reach target conditions. This category consists of one skill, Feedback-Based Device Control.

The Feedback-Based Device Control skill communicates with the corresponding device upon receiving a Heat or Stir command and controls execution until the target physical conditions are met, thereby preventing reaction failures caused by premature progression. Internal sensor values such as temperature or RPM, streamed from the Perception Module (Section III-C3), are monitored, and execution is suspended or locally controlled until the target conditions are met, ensuring that experimental conditions are enforced during execution rather than assumed during planning.

IV. EXPERIMENTS AND EVALUATION

This section evaluates the framework in the simulation testbed (Fig. 6) against the three limitations of Section II-D, proceeding from XDL generation, planning reliability and latency, and tool selection and rearrangement reasoning to end-to-end performance on long-horizon chemical workflows. Finally, the pouring skill and the Transfer procedure are validated on the physical FR5. Demonstration videos are provided on the project page at https://rcilab.khu.ac.kr/auto_chem/.

A. EXPERIMENT SETUP

This section describes the hardware platform and simulation environment, the chemical tasks selected to span a range of manipulation difficulty, and the fine-tuning procedure for the two LLM components.

1) Hardware, Simulation, and Planner Configurations

The proposed framework is evaluated on a 6-DoF collaborative manipulator (FAIRINO-FR5, Fig. 6) equipped with three end-effectors to cover the range of object geometries encountered in chemistry laboratory workflows: the 2-finger gripper (AG95) for Transfer, which grips a vessel from the

side and tilts it to pour, the 3-finger gripper (*DH3*) for holding cylindrical vessels stably from above, and the suction gripper (*VGC10*) for objects with a flat top surface held by suction, such as capped bottles, boxes, and trays. As illustrated in Fig. 13, no single gripper configuration can reliably cover all object geometries, motivating the LLM-based dynamic tool assignment described in Section III-B3.

A high-fidelity simulation testbed was developed within *NVIDIA Isaac Sim* 6.0.0.1, generating 30 distinct initial configurations per task under identical constraint settings to support large-scale randomized trials and controlled ablation. The TAMP World State is built at each stage from the object poses and the joint configuration of the robot, and the optimized trajectories are streamed back to the simulation for execution. The object poses come from one of two sources. The ground-truth state uses the poses that the simulator provides. The perception state uses the 6-DoF poses that the Perception Module estimates from the rendered images of the two fixed cameras and the wrist camera. The workflows of Section IV-E are evaluated in both states from the same initial configurations. The simulation environment and the generation of configurations are summarized in Table 9 of Appendix A, and the settings of the two planners used in the planner comparison of Section IV-C in Table 10. The PDDLStream attempt limits for placement poses and inverse kinematics were set to the values with the highest success rate on ten initial configurations held out from the evaluation. Execution in simulation includes three sources of uncertainty: the initial configuration and arm configuration drawn at random for each trial (Table 9), the pose estimation error of the Perception Module that enters the World State in the perception state, and the grasp and placement outcomes that follow from the rigid-body contact dynamics of the physics engine. The flow of liquids and powders and the scale measurement are not simulated. The simulated Transfer is therefore judged only up to the execution of the pouring motion, and pouring accuracy is evaluated in the physical experiments of Section IV-F.

2) Chemical Experiments

Three manipulation tasks are selected to cover the essential procedures of real-world chemical experiments while spanning a range of planning complexity and manipulation difficulty. Move represents pick-and-place of laboratory objects and is evaluated using objects held from the top by the suction gripper to assess the LLM's tool selection capability and manipulation success rate. Transfer represents liquid handling, a fundamental procedure in chemical synthesis, and tests the system's ability to maintain spill-free orientation constraints throughout the transit phase while tilting the source container and pouring its contents into a target vessel. Stir is chosen over simpler device operations such as heating because it demands higher spatial precision: the robot must relocate the vessel onto a magnetic stirrer and precisely insert a magnetic stir bar into the narrow opening, making it an ideal scenario for evaluating complex, multi-step task and motion planning.

A trial counts as a planning success when a collision-free

trajectory that satisfies the upright bound $\theta_{\max}$ is returned within the time budget. It counts as an execution success when, after a settling period of 2 s, the plan has been carried out in full and the target rests upright inside its goal region. The full criteria for each task are given in Appendix A.

3) LLM Fine-Tuning and Training Details

The two LLM components introduced in Section III-B, the XDL Generator and the Action Reasoner, are each fine-tuned on a domain-specific dataset constructed to reflect the combinatorial structure of the target task. For the XDL Generator, the training dataset is constructed at a scale of 5,000 samples across four combinatorial axes: synonym variations for each of the six core XDL operators (Add, Stir, HeatChill, Transfer, CleanVessel, Move), continuous numerical parameter ranges (0.5–20 mL in steps of 0.5 mL for volume, 0–20 °C for temperature, 1–30 s for time), three instance identifiers for each of four vessel types, yielding 12 unique object identifiers, and command complexity ranging from single-operator instructions to three-step composite procedures with a weighted distribution of 30% single-step, 40% two-step, and 30% three-step. Since the task constitutes a structural mapping to a fixed XDL schema, the output space is inherently constrained and this dataset covers the principal combinatorial patterns, as supported by the results in Section IV-B1. The training, validation, and evaluation commands are separated at five levels, namely sentence form, wording, object, composition, and numerical value, as detailed in Appendix B, and a validation set of 500 commands is used to monitor training. Training is performed by applying LoRA (rank $r = 16$, $\alpha = 32$, dropout 0.05) to a 4-bit quantized *Llama 3.2 1B* model, using an 8-bit AdamW optimizer with a learning rate of 2×10^{-4} , an effective batch size of 16, and 5 epochs.

For the Action Reasoner, the training dataset consists of 3,000 samples generated combinatorially by placing up to four objects on 12 workspace grids. For each combination of the six XDL operators, rearrangement necessity is determined according to task-specific blocking criteria, and the ground-truth target grid is assigned as either a context-aware target zone or a workspace edge depending on whether the obstacle is the target of the subsequent procedure. Candidate workspace grids are provided as input features after excluding fixed equipment and occupied grids, categorized solely by locational characteristics to prevent label leakage. The dataset is balanced across rearrangement-required and rearrangement-unnecessary cases, as well as context-aware and irrelevant obstacle cases, to prevent over-prediction of rearrangement. Training and evaluation scenarios are separated along three axes, namely object, spatial relation, and task combination, as detailed in Appendix B. Training follows the same *Llama 3.2 1B* base model with LoRA configured at rank $r = 8$, $\alpha = 16$, dropout 0.05, a learning rate of 1×10^{-4} , batch size 16, and 3 epochs. The differing LoRA configurations were selected empirically based on validation performance, rather than through a dedicated hyperparameter ablation.

TABLE 1. Field-level accuracy of the XDL Generator over the 2,300 commands of the seven evaluation sets. The matched control follows the training distribution, the next five sets each differ from it at one separation level, and the last differs at all five. Operators, procedural ordering, and exact match are proportions of the n protocols, and a protocol that cannot be read as XDL counts as a mismatch in all three. Argument names, object identities, numerical values, and units are proportions of the fields in the paired steps of the readable protocols. Values are percentages, with a 95% confidence interval on exact match.

Evaluation set	n	Operators	Argument names	Object identities	Numerical values	Units	Procedural ordering	Exact match [95% CI]
Matched control	500	100.0	100.0	99.9	100.0	100.0	100.0	99.8 [98.9, 100.0]
Unseen object	300	100.0	100.0	100.0	100.0	100.0	100.0	100.0 [98.7, 100.0]
Unseen numerical value	300	100.0	100.0	100.0	100.0	100.0	100.0	100.0 [98.7, 100.0]
Unseen composition	300	99.7	100.0	100.0	100.0	100.0	99.7	99.7 [98.1, 99.9]
Unseen sentence form	300	98.3	100.0	100.0	100.0	100.0	98.3	98.3 [96.2, 99.3]
Unseen wording	300	98.3	100.0	99.8	100.0	100.0	99.7	97.7 [95.3, 98.9]
All five at once	300	95.3	100.0	99.9	99.3	100.0	97.7	93.3 [89.9, 95.6]

TABLE 2. Outcome of the XDL Validator on the 925 benchmark protocols, on the 8,100 reference protocols of the generator data, and on the errors the generator produced on its own.

Protocol set	n	Outcome (%) [95% CI]
Must be rejected, syntactic	250	100.0 [98.5, 100.0] rejected
Must be rejected, object	150	100.0 [97.5, 100.0] rejected
Must be rejected, feasibility	225	100.0 [98.3, 100.0] rejected
Must pass, ordinary	200	0.0 [0.0, 1.9] rejected
Must pass, near-miss	100	0.0 [0.0, 3.7] rejected
Must pass, reference protocols	8,100	0.0 [0.0, 0.05] rejected
Errors made by the generator	34	88.2 [73.4, 95.3] rejected

B. XDL GENERATION EVALUATION

This section evaluates the XDL generation pipeline across two aspects: the accuracy of the XDL Generator in converting natural language commands into structured protocols, and the ability of the XDL Validator to detect and intercept physically invalid protocols before execution.

1) Language Command to XDL Format

To assess whether the XDL Generator reliably converts natural language commands into structured protocols across varying task complexities, 2,300 test commands spanning single-step to three-step composite workflows were evaluated over seven evaluation sets. To separate generator accuracy from validator performance, each protocol was scored against a reference protocol rather than by the validator.

Each procedure was defined as a step list before any model was evaluated, and the natural-language command and the reference protocol were then written separately from that same list. A quantity is held as a number and a unit apart from each other, so that “5 mL” is stored as the value 5.0 and the unit mL rather than as a string. Before comparison, vessel names were case-folded, volumes were expressed in mL, temperatures in °C, and times in s, and the data type of every attribute was checked against the XDL schema.

Operators were judged at the level of the whole protocol, by whether the same names occupy the same positions as in the reference. The steps of the two protocols were then paired in order by aligning the two operator sequences with the fewest insertions, deletions, and substitutions. Procedural ordering is the proportion of protocols in which every reference step is paired with a generated step. Argument names, object identities, numerical values, and units were judged

within each paired step that has the same operator as the reference. A protocol that cannot be read as XDL counts as a mismatch in operators, procedural ordering, and exact match, and contributes no fields to the other four. A protocol counts as an exact match only when every field agrees.

Table 1 reports the outcome over the 2,300 commands of the seven evaluation sets. Only 34 protocols differ from their reference. Argument names and units agree in every set, and object identities and numerical values agree in at least 99.3% of the fields of the readable protocols. No protocol contains the right operators in the wrong order. Of the 34, 14 cannot be read as XDL, 11 perform an unrequested operation, 8 by replacing a step and 3 by adding one, and the other 9 bind a wrong object or value in an otherwise correct protocol.

Operator accuracy is 100.0% on the sets that vary only the object or the numerical value, falls to 98.3% on the sets built from unseen wording or unseen sentence forms, and falls to 95.3% on the set that varies all five levels at once. Decoding is not grammar-constrained, so the model can write output that is not valid XDL. Of the 14 unreadable protocols, 5 use a tag outside the six operators, in each case by copying a verb of the command into the tag name, and 9 place a range such as “0.5–20 mL” or a name where a quantity belongs. All 5 errors on unseen sentence forms are of the second kind, and they lower operator accuracy and procedural ordering on that set together. The median inference time per command is 0.41–0.65 s across the evaluation sets and the slowest five percent stay within 0.77 s, confirming that the module introduces negligible latency relative to the overall pipeline execution time.

2) XDL Validator

To measure how accurately the XDL Validator detects the error types it defines in advance, a benchmark of 925 protocols was built over the 25 conditions listed in Table 11. Of these, 625 are error protocols designed to violate a single condition each, 25 per condition, and have to be rejected. The remaining 300 violate no condition and have to pass. Of those 300, 200 are ordinary procedures and 100 are constructed to sit exactly on a boundary condition, such as a vessel capacity, a remaining volume, or the reuse of a vessel that still holds the same reagent, without violating it.

Table 2 reports the outcome. Every protocol that has to

TABLE 3. Comparison of planning success rate and time to solution between TAMP solvers based on sequential parameter binding (PDDLStream) and batched joint optimization (cuTAMP) on the 6-DoF *FR5* manipulator. Each task was evaluated on the same 30 randomized initial configurations within a 60 s budget. Success rates are given with Wilson 95% confidence intervals and the exact McNemar test p -value on the paired configurations. Trials that return no plan within the time budget, including both timeouts and early terminations, are right-censored at 60 s. The restricted mean time to solution (RMTS) is the mean time computed by assigning the budget (60 s) to unsolved trials, and planning latency is given over successful trials only as the median [interquartile range, IQR].

Task	Solver	Success (%) [95% CI]	McNemar p	RMTS (s)	Planning time (s) median [IQR]	Timeout (%)	Early termination (%)
Transfer	PDDLStream	73.3 [55.6, 85.8]	7.8×10^{-3}	21.3	0.97 [0.60, 3.34]	26.7	0.0
	cuTAMP	100.0 [88.6, 100.0]		15.9	14.65 [14.17, 14.90]	0.0	0.0
Move	PDDLStream	100.0 [88.6, 100.0]	1	0.1	0.08 [0.06, 0.10]	0.0	0.0
	cuTAMP	100.0 [88.6, 100.0]		8.9	8.89 [8.87, 8.92]	0.0	0.0
Stir	PDDLStream	100.0 [88.6, 100.0]	1	1.8	0.83 [0.51, 2.23]	0.0	0.0
	cuTAMP	100.0 [88.6, 100.0]		12.8	12.00 [11.84, 12.09]	0.0	0.0

be rejected is detected without exception, and not one of the 300 that have to pass is rejected, so no false positive occurs. The validator also accepted all 8,100 reference protocols of the training, validation, development, and evaluation data together with their commands. This establishes that the detection rate is not obtained by over-rejecting borderline cases. Enabling the three validation stages in turn, namely syntactic, object-existence, and physical-feasibility validation, raises the cumulative detection over the 625 error protocols from 40.0% to 64.0% and then to 100.0%. Each error condition is first detected when its own stage is enabled and is not caught by the stages before it, and the three stages combined detect all 25 defined conditions.

Detection on injected errors alone does not fully establish how far the validator protects the pipeline in practice, because the injection procedure and the validation logic share the same rules and the evaluation can therefore be circular. The 34 incorrect protocols that the XDL Generator actually produced in Section IV-B1 are free of that circularity. The XDL Validator rejected 30 of them, a blocking rate of 88.2% [95% CI: 73.4, 95.3]. Checking the same 34 protocols without cross-referencing the natural-language command lowers the rejection rate to 67.6%, which indicates that resolving the quantities and object names of the command against the protocol accounts for 7 of the 30 rejections on its own. The 4 that were not rejected are valid XML, reference only objects in the World State, satisfy every operator precondition, and use exactly the quantities and names of the command, but select an operation the command did not ask for.

Finally, the validator was evaluated on 200 anomalous commands for which no physically valid protocol can exist at all. These are illogical procedures in which a later operator requires a vessel state that an earlier operation has already invalidated. The validation pipeline rejected 100.0% [95% CI: 98.1, 100.0] of them and returned the expected error type in every case. On each rejection the system returned an error message identifying the cause, such as “Cannot Stir empty vessel: beaker_A,” so that the user can revise the instruction before any robot motion is performed. These quantitative results establish the discriminative capability of the validator over the 25 defined error conditions.

C. TAMP PLANNING RELIABILITY AND LATENCY EVALUATION

This section compares the planning reliability and termination time of the proposed planning module with PDDLStream, a representative sequential parameter-binding baseline (Section II-B), on the three tasks with the same 30 initial configurations per task, goal conditions, and orientation constraints. The settings of the two planners are given in Table 10. To compare the planning modules themselves, both planners use the object poses provided by the simulator. Both planners run on the same workstation with the same 60 s budget, each on the compute device its public implementation uses, so Table 3 compares them on the same hardware and the same wall-clock time. The reported metrics reflect algorithmic planning latency only, excluding physical execution and simulation rendering time, and include restarts within the budget.

The proposed planning module returns a plan within the budget for every configuration of the three tasks. The difference in success rate appears on the Transfer task, where orientation constraints are most restrictive. On the same 6-DoF manipulator the two planners reach 100.0% [88.6, 100.0] and 73.3% [55.6, 85.8], and the difference is significant on the paired configurations ($p = 7.8 \times 10^{-3}$, exact McNemar). On Move and Stir both planners solve every configuration. Both planners were run to 180 s, and the success counts and McNemar p -values are the same at budgets of 60, 120, and 180 s. Beyond success rates, the time of one optimization round of the proposed planning module falls within a narrow range on each task, and it solves 86 of the 90 configurations of the three tasks in the first round. The results are summarized in Table 3 and Fig. 7.

On the Transfer task, the proposed planning module solves every configuration by jointly optimizing candidate particles in parallel toward the constraint manifold, which enables effective exploration of the feasible region even when kinematic redundancy is limited. PDDLStream, by contrast, stops at 73.3%, and all of its failures (26.7%) are timeouts, as its sequential sampling mechanism depends heavily on whether initial random configurations happen to satisfy the orientation constraints. On Move and Stir the feasible region is wide enough for independently drawn samples to satisfy the constraints together. These results demonstrate that

TABLE 4. Action Reasoner rearrangement decision accuracy per evaluation set. Each set differs from the training distribution along one axis. Values are percentages. Rearrange needed and the auxiliary tool are scored on all n scenarios of each set, where the correct auxiliary tool is None when no rearrangement is needed. The target grid is scored on the n_d scenarios of each set that require a rearrangement, with a 95% confidence interval.

Evaluation set	n	Rearrange needed	Auxiliary tool	n_d	Target grid
Matched control	400	100.0	100.0	151	88.7 [82.7, 92.8]
Unseen instance name	300	100.0	100.0	164	95.1 [90.7, 97.5]
Unseen object type	300	100.0	93.0	180	92.8 [88.0, 95.7]
Unseen spatial relation	300	100.0	100.0	106	84.9 [76.9, 90.5]
Unseen task combination	300	100.0	100.0	143	94.4 [89.3, 97.1]

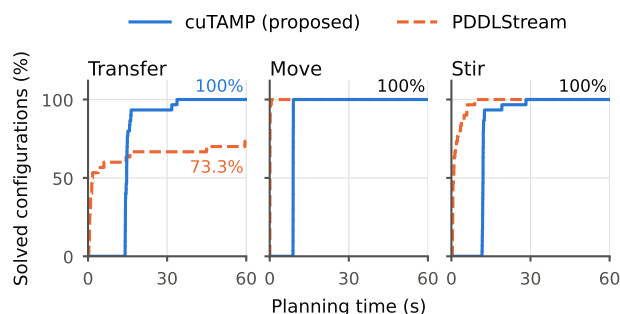


FIGURE 7. Cumulative solved fraction of the two planners against wall-clock time (Kaplan-Meier curves), over the same 30 randomized initial configurations per task. A trial that returns no plan within the 60 s budget is right-censored at that point and stays in the denominator, so the final height of each curve gives the final success rate of that planner.

batched joint differentiable optimization handles orientation constraints more reliably than sequential parameter binding on low-redundancy manipulators when the constraint manifold is narrow.

The two approaches also differ in temporal predictability. Fig. 7 plots the solved fraction against wall-clock time, with trials that return no plan within the budget censored at 60 s. On Transfer the baseline quickly solves the configurations whose initial samples satisfy the constraints, but cannot solve the others before the budget runs out, so its curve stops at 73.3%. The proposed planning module, by contrast, solves every configuration of each task within the budget, and its solve times gather by optimization round. On Transfer, 28 of the 30 configurations were solved in the first round at 14.1–16.3 s, and the other 2 after one restart at 31.8 s and 33.7 s. The restricted mean time to solution (RMTS), computed by assigning the budget to unsolved trials, is 15.9 s for the proposed planning module and 21.3 s for the baseline on Transfer. Overall, unlike PDDLStream, the proposed planning module consistently terminates within a time proportional to the number of rounds while returning a valid solution, a property that is critical for time-sensitive chemical workflows in which unpredictable planning delays may compromise reaction conditions.

D. LLM-BASED TOOL SELECTION AND REARRANGEMENT ACTION REASONING

To verify that the LLM Module reliably performs its two core functions, namely semantic tool assignment and context-aware obstacle rearrangement, the module is evaluated on

unseen test scenarios not present in the fine-tuning dataset. The evaluation is structured as a two-stage ablation, because both functions are exercised within a single action sequence and removing the module entirely would make it difficult to distinguish their individual contributions. Section IV-D1 evaluates the inference accuracy of tool assignment and of the rearrangement decisions on unseen scenarios. Section IV-D2, in contrast, isolates the rearrangement strategy by holding tool selection constant and comparing context-aware placement against no-rearrangement and random-rearrangement baselines. Section IV-E then evaluates the two functions together in the integrated pipeline.

1) Semantic Tool Assignment and Action Reasoner Evaluation

To assess whether the LLM Module assigns appropriate end-effectors across diverse task–object combinations, including object types not encountered during fine-tuning, tool selection accuracy is evaluated on 1,600 unseen test scenarios spanning the six XDL operators and the three end-effectors. Table 12 reports the results by task and target object type. The module reaches 100.0% accuracy in every task–tool condition, including the object types absent from the fine-tuning data. The model consistently assigns the 3-finger gripper for cylindrical vessels across Move and Stir tasks, the 2-finger gripper for Transfer to enable side-grasp pouring, and the suction gripper for flat-surface objects that exceed the graspable width of finger-based end-effectors. The average inference time of 0.083 s confirms that tool selection introduces negligible latency into the overall pipeline.

Because no single end-effector covers all task–object combinations (Fig. 13), this accuracy also supports dynamic tool assignment as a prerequisite for covering them.

Beyond tool selection, the overall reasoning of the Action Reasoner is evaluated over all four of its outputs: the main tool, rearrangement necessity, the auxiliary tool, and the target grid. Following the main tool above, Table 4 summarizes the remaining three outputs, which govern the rearrangement action. The evaluation covers a matched control and four datasets that differ from the training distribution. Each of these differs from the training distribution along only one of the three axes of object, spatial relation, and task combination, with the object axis split into two sets of unseen instance names and unseen object types, so that a change in performance can be attributed to that factor.

Rearrangement necessity is decided correctly in 100.0% of

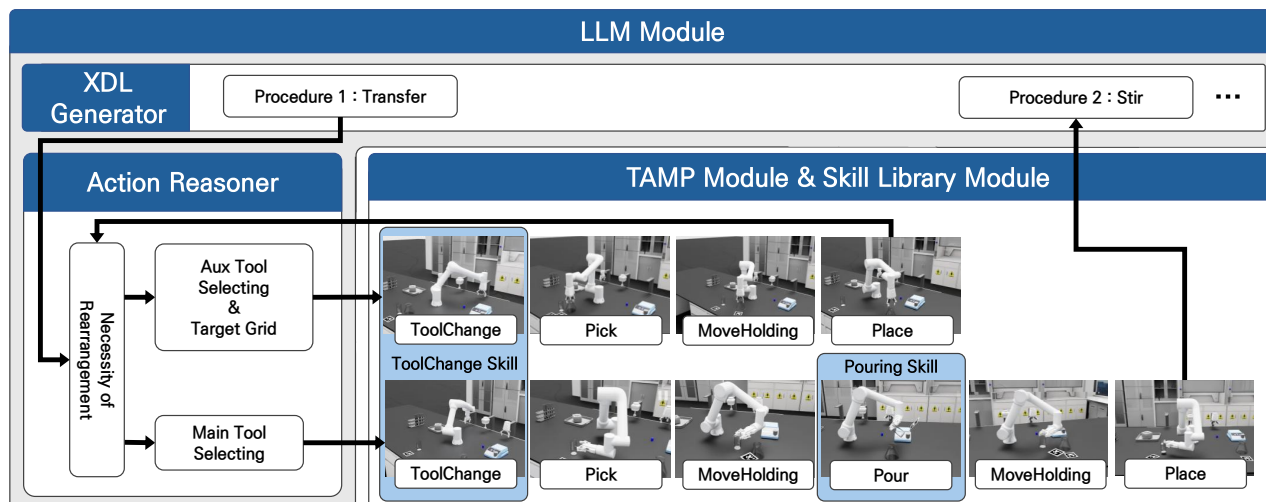


FIGURE 8. Execution flow of the TAMP pipeline for a Transfer procedure. The Action Reasoner determines tool selection and rearrangement necessity. When rearrangement is required, an auxiliary rearrangement sequence (upper path) is inserted prior to the main experimental procedure (lower path), which includes sensor-feedback-based skills such as adaptive pouring.

TABLE 5. TAMP success rates and sequence completion rates by rearrangement strategy. Each sequence-condition pair is evaluated over 30 randomized trials. Values are percentages.

Sequence	Metric	None	Random	Context-aware
Transfer → Stir	Step 1 success	0.0	60.0	96.67
	Sequence completion	0.0	50.0	96.67
Stir → Transfer	Step 1 success	100.0	100.0	100.0
	Sequence completion	13.33	40.0	83.33

cases under every condition. The auxiliary tool reaches 93.0% on the unseen object type condition and 100.0% elsewhere, because an obstacle has to be identified from the scene description, unlike the main task object, which is given as an XDL argument. The target grid, scored by whether the named grid lies within the admissible zone, is lowest at 84.9% on the unseen spatial relation condition.

2) Context-Aware Rearrangement Strategy Evaluation

To isolate the contribution of procedural context in rearrangement decisions, three conditions are compared. The No Rearrange condition executes the main task directly even when obstacles block the workspace. The Random Rearrange condition detects obstacles and relocates them to arbitrarily selected unoccupied grids, serving as an ablation that preserves spatial clearance but removes context-awareness. The Context-Aware condition uses the Action Reasoner to analyze the subsequent procedure and preemptively relocate obstacles to the work zone required by the next step. Tool selection is held constant across all three conditions to isolate rearrangement strategy as the sole independent variable.

Because the benefit of procedural context arises at each transition from one procedure to the next, the rearrangement strategies are first compared over all 34 transitions in which a next procedure follows. Forty situations were built for each transition, giving 1,360 in total, and in each the object that has to be moved out of the way is the object the next pro-

cedure acts on. Over the 400 situations in which the next procedure requires a particular station, the mean number of manipulations falls from 2.078 under random rearrangement to 1.298 under context-aware rearrangement, a reduction of 37.5%. Context-aware rearrangement puts the object on that station in 87.0% of situations against 9.0% for random rearrangement, so the object does not have to be moved again in the next procedure. Over the remaining 960 situations, in which the next procedure requires no particular position, the mean numbers of manipulations are 1.053 and 1.058, so context-aware rearrangement stays at the level of random rearrangement without degradation. The benefit of context is thus concentrated in transitions where the next procedure requires a particular position.

To see what difference the rearrangement strategy makes to actual planning and execution in such transitions, Table 5 reports the Step 1 TAMP success rate and sequence completion rate for each condition across two evaluation sequences: Transfer → Stir, in which the next procedure requires the stirrer station and the wide workspace required by side-grasp pouring makes rearrangement most frequent, and the reverse order Stir → Transfer as a control. Context-aware rearrangement achieves the highest sequence completion rate in both cases, with 96.67% and 83.33% against 50.0% and 40.0% for random rearrangement and 0% and 13.33% without it.

Without rearrangement, workspace congestion blocks Step 1 of Transfer → Stir. Random rearrangement clears the immediate workspace, but because obstacles are placed without regard to the subsequent procedure, Step 2 encounters new collisions or requires additional rearrangement. Context-aware rearrangement secures the workspace for Step 1 and places obstacles where Step 2 needs them, eliminating the redundant manipulations of random placement. Together with the evaluation over all transitions, these results confirm that procedural context, rather than spatial clearance alone, is the decisive factor when the next procedure requires a particular

TABLE 6. Number of successful trials at each stage and End-to-End success rate for the long-horizon chemical workflows. Each workflow was evaluated from the same 30 initial configurations in the ground-truth state and in the perception state. Brackets give Wilson 95% confidence intervals.

Experimental Workflow	State	XDL Generation	Perception	Planning	Execution	End-to-End Success (%)	McNemar p
Move → Transfer	Ground truth	30/30	–	30/30	29/30	96.7 [83.3, 99.4]	1.0
	Perception	30/30	30/30	30/30	30/30	100 [88.6, 100]	
Transfer → Stir → Move	Ground truth	30/30	–	30/30	29/30	96.7 [83.3, 99.4]	1.0
	Perception	30/30	30/30	30/30	30/30	100 [88.6, 100]	
Move → Transfer → Stir	Ground truth	30/30	–	30/30	30/30	100 [88.6, 100]	1.0
	Perception	30/30	29/30	29/29	29/29	96.7 [83.3, 99.4]	
Total	Ground truth	90/90	–	90/90	88/90	97.8	
	Perception	90/90	89/90	89/89	89/89	98.9	

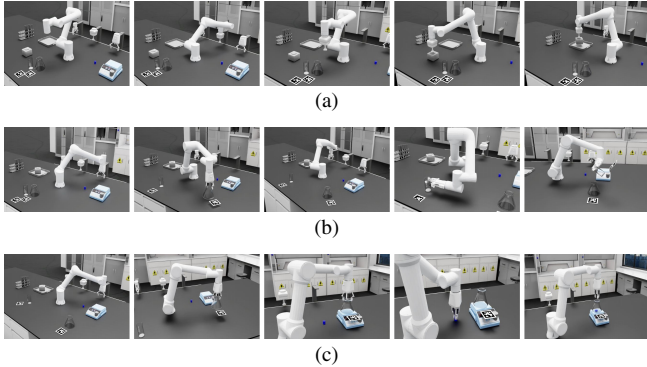


FIGURE 9. Sequential execution snapshots of the Move → Transfer → Stir workflow evaluated in the end-to-end experiments. Each row shows five representative keyframes captured during a single trial. (a) Move: the robot relocates a capped bottle to the target zone. (b) Transfer: liquid reagent is poured into the destination vessel. (c) Stir: the mixture is stirred on the hotplate to complete the reaction sequence.

position, and that it causes no degradation when it does not. Fig. 8 shows how the rearrangement sequence is inserted before the main task of a Transfer procedure.

E. END-TO-END SYSTEM EVALUATION

While the preceding sections have verified planning reliability (Section IV-C) and operational flexibility (Section IV-D) independently, this section evaluates whether these capabilities are preserved when the modules operate as an integrated pipeline. A trial starts from a single natural-language command and counts as a success only when every procedural step, in order, meets the execution-success criteria of Section IV-A2. The success rates of the planning and execution stages also follow the definitions of planning success and execution success in that section. In the perception state, a perception stage precedes the planning of each procedure, and it counts as a success when the positions of the target objects are fixed. The ground-truth state uses the poses of the simulator and has no such stage. A trial counts as failed at the first stage it fails, and the success rate of each stage is taken over the trials that reached that stage. The End-to-End success rate is taken over all trials. The End-to-End outcomes of the two states are compared with the exact McNemar test, pairing trials from the same initial configurations. Each workflow is evaluated over 30 independent trials initialized with randomized World States and robot configurations.

TABLE 7. Localization error of the object positions estimated in the perception state just before the planning of each procedure. All measurements of the 237 procedures are included, and values are mean / 95th percentile.

Object	Horizontal position (mm)	Height (mm)	Rotation ($^{\circ}$)
Beaker	3.06 / 6.74	2.38 / 6.69	0.09 / 0.22
Flask	2.94 / 5.78	4.09 / 6.65	0.09 / 0.24

To identify bottlenecks, end-to-end performance is decomposed into the stages of XDL generation, planning, and execution, and in the perception state the perception stage defined above is added before planning. As shown in Table 6, XDL generation succeeded in every trial in both states. The End-to-End success rate is 97.8% (88/90) in the ground-truth state and 98.9% (89/90) in the perception state, and the difference between the two states is not significant in any of the three workflows (McNemar $p = 1.0$). The two failures in the ground-truth state occurred in the same initial configuration, where the beaker set down after Transfer tipped over, and they arose from rigid-body contact. The one failure in the perception state occurred in a configuration where part of a marker board was occluded by another board. The wrist camera went through all 11 viewpoints without finding the beaker, so execution stopped and the object was reported, as designed in Section III-C1. Every trial that passed perception succeeded in planning and execution.

For the 237 procedures in the perception state, the position estimated just before the planning of each procedure was compared with the true position in the simulator (Table 7). Object positions were fixed by the fixed cameras 470 times, and the wrist-camera search was used 4 times. The horizontal position error was 3 mm on average and within 7 mm at the 95th percentile. In 1 of the 474 measurements, the tilt of the marker board was misjudged and the flask position was off by 65 mm, yet the procedure was completed. Excluding it, the maximum error was 9.7 mm horizontally and 9.6 mm in height. No trial failed from pose estimation error.

Fig. 9 shows a representative trial of the Move → Transfer → Stir workflow. The success rates in the two states show that the planning reliability and operational flexibility of the preceding sections are preserved under integrated multi-step execution on the perceived state.

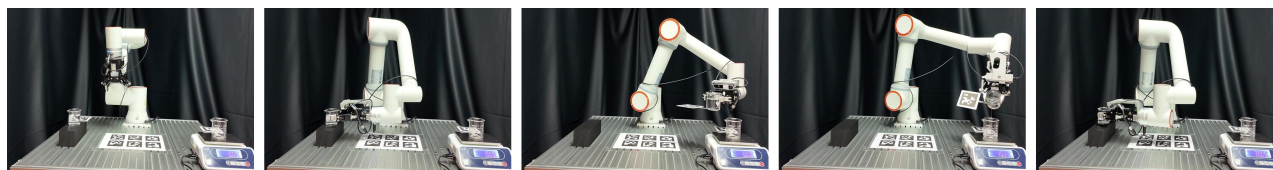


FIGURE 10. Keyframes of the Transfer procedure performed on the physical FR5. The robot picks up the source beaker, carries it upright, pours into the beaker on the scale, and returns it to its place.

TABLE 8. Pouring accuracy on the physical FR5. Each reagent and target mass was poured 10 times. The error is the difference between the scale reading 3 s after the pour ends and the target mass, and proportions are given with Wilson 95% confidence intervals.

Reagent	Target (g)	Within ± 1.0 g (%)	Within ± 0.5 g (%)	Error (g) mean $\pm$ SD	Mean absolute error (g)	Duration (s)
Water	10	100 [72, 100]	80 [49, 94]	-0.02 ± 0.37	0.26	40–66
	30	100 [72, 100]	90 [60, 98]	$+0.16 \pm 0.23$	0.23	42–51
	50	100 [72, 100]	80 [49, 94]	$+0.22 \pm 0.30$	0.32	46–51
Oligosaccharide	10	100 [72, 100]	70 [40, 89]	$+0.18 \pm 0.38$	0.34	52–86
	30	100 [72, 100]	90 [60, 98]	$+0.12 \pm 0.24$	0.19	56–74
	50	100 [72, 100]	100 [72, 100]	$+0.14 \pm 0.20$	0.21	69–75
Powdered sugar	10	80 [49, 94]	60 [31, 83]	$+0.37 \pm 0.96$	0.64	42–79
	30	90 [60, 98]	70 [40, 89]	$+0.27 \pm 0.52$	0.36	52–89
	50	60 [31, 83]	40 [17, 69]	$+0.65 \pm 0.62$	0.73	45–82

F. VALIDATION ON THE PHYSICAL FR5

To validate the execution layer and the Perception Module under the flow of real reagents and real sensor characteristics, three experiments were carried out on the physical FR5. The physical setup consists of the FR5 of Fig. 6 with the 2-finger gripper, the two fixed cameras and the wrist camera, and an electronic scale on which the receiving beaker stands. The scale measures weight at a resolution of 0.01 g and a rate of 5 Hz, and the flow rate is estimated from the readings of the most recent 0.6 s, three scale measurement periods. Three reagents were used, water, an oligosaccharide syrup with a viscosity of about 3,000 cP at 26 °C, and powdered sugar, representing a low-viscosity liquid, a high-viscosity liquid, and a granular reagent.

1) Pouring Accuracy

Pouring accuracy was measured without picking and carrying, by tilting the source beaker held in the gripper from a fixed start pose. Each reagent was poured 10 times at each of the target masses 10, 30, and 50 g, and the error is defined as the difference between the scale reading 3 s after the pour ends and the target mass.

Table 8 reports the results per reagent and target mass. Water and the oligosaccharide syrup reached within ± 1.0 g of the target in all 60 pours, with a mean absolute error of 0.19–0.34 g and a maximum absolute error of 0.77 g. The syrup is about 3,000 times more viscous than water yet reached the same accuracy, and because it flows more slowly a single pour took 52–86 s against 40–66 s for water. This is because the pouring skill estimates the flow rate directly from the scale readings, so the difference in flow rate due to viscosity is carried straight into the tilt adjustment. Powdered sugar, in contrast, reached within ± 1.0 g in 23 of 30 pours, and its mean error was positive at every target mass, biased toward overshoot. Powder does not flow continuously but slides down in clumps, so the amount that has already begun to slide when

the stop is decided is added to the pour.

2) Physical Execution of the Transfer Procedure

The Transfer procedure was then executed on the physical robot. In each run the planning module planned the trajectory from the positions of the two beakers measured by the Perception Module, and the robot followed this trajectory to pick up the source beaker, carry it upright, pour the target mass with the pouring skill, and return the beaker to its place. One run was made for each of the nine combinations of the three reagents and the target masses 10, 30, and 50 g, and the source beaker was moved between runs so that its measured position differed by up to 65 mm. All nine runs completed the procedure, and a representative run is shown in Fig. 10. The trajectories excluding the pour were 37–45 s long and were executed as planned without any position correction, which shows that the position estimates of the Perception Module are accurate enough for grasping and for aligning the pour. The planning module plans the collision-free trajectory to the pouring pose and the nominal tilting trajectory, and the pouring skill corrects this trajectory in real time with scale feedback and carries out the quantitative transfer. In all nine runs the pouring skill took over the transfer from the planned start pose, which confirms that this division of labor also works on the physical robot.

3) Marker Search with the Wrist Camera

Finally, the marker search with the wrist camera was checked on the physical robot. In the 30 g water run, the tag of the receiving beaker appeared small in the fixed-camera images and fell short of the measurement quality criterion, so the arm carried the wrist camera through 36 predefined viewpoints. At the 21st viewpoint the wrist camera read the tag and updated the position of the beaker, and the Transfer was executed with the trajectory planned from that position (Fig. 11).

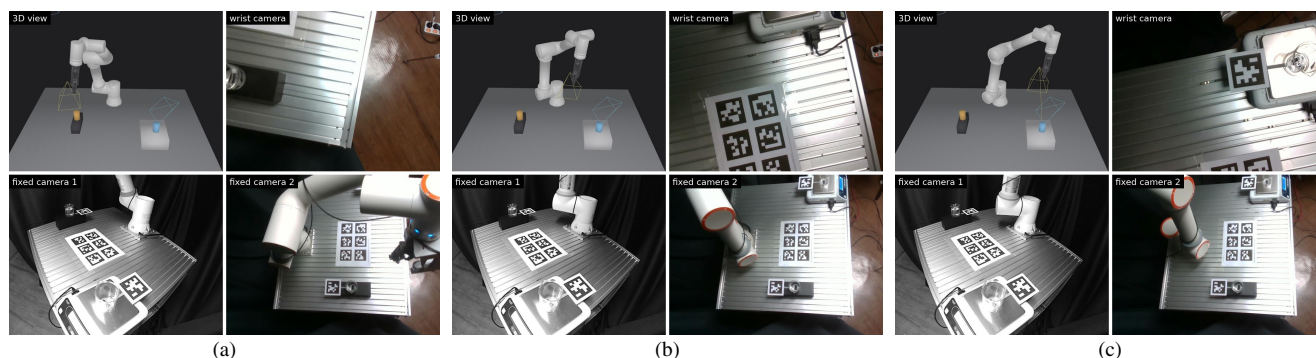


FIGURE 11. Marker search with the wrist camera on the physical FR5. Each panel shows, at the same instant, a 3D reconstruction (upper left), the wrist camera (upper right), and the two fixed cameras (bottom). The reconstruction is drawn from the recorded joint states and the perceived beaker positions. (a) Start of the search. (b) During the search. (c) The moment the tag of the receiving beaker is read at the 21st viewpoint and its position is updated.

V. DISCUSSION AND LIMITATIONS

This section discusses the assumptions and limitations that bound the interpretation of the results in Section IV.

A. SCOPE OF THE PROCEDURES EVALUATED IN SIMULATION

The physical experiments of Section IV-F cover pouring and the Transfer procedure. The other procedures were evaluated in simulation, which reproduces physics only up to rigid-body contact dynamics (Section IV-A1). A procedure whose success depends on physical processes beyond rigid-body contact therefore cannot be judged from simulation alone.

In simulation the tool is swapped as soon as the arm reaches the exchange point, without the electromagnet control and the latching motion of Section III-E1, so coupling failures do not appear in the results. Execution success of Stir is judged by whether the stir bar rests inside the mouth of the flask (Section IV-A2), whereas whether a solute actually dissolves depends on its solubility, the temperature, and the stirring time, which the simulation does not reproduce. Success of Stir therefore means that a placement from which stirring can start has been reached.

B. PERCEPTION ROBUSTNESS UNDER REALISTIC LABORATORY CONDITIONS

The Perception Module assumes that fiducial markers remain detectable throughout operation. In a working chemistry laboratory this assumption can be violated: markers may degrade through chemical spills, condensation, or thermal exposure near hotplates, and may be transiently occluded by the manipulator or by relocated vessels. The current design provides two layers of tolerance. First, the two fixed cameras maintain tracking continuity when one view is occluded, since the remaining valid view is used directly when fusion is not possible, and the wrist camera searches for the marker when neither view can read it or its measurement cannot be trusted (Section III-C1). Second, because the World State is grounded at procedure boundaries rather than continuously during motion (Section III-C2), transient detection losses between stages do not corrupt planning inputs: the most recently

verified pose remains valid, and objects do not move between manipulations by construction of the workflow.

These mechanisms do not cover permanent marker loss. Operationally, the framework assumes laminated markers mounted away from heated surfaces and periodic recalibration of the fixed camera extrinsics. The fixed tag-to-object offset assumes rigid vessels with known marker placement and requires re-registration whenever markers are re-affixed. Because the sphere covering of Section III-C2 is conservative, approximation errors in the collision geometry bias the planner toward rejecting marginal plans rather than toward collision, at the cost of occasional infeasibility in tightly cluttered scenes. Replacing fiducial tracking with marker-free 6-DoF pose estimation, adapting the estimator continually as the appearance of the laboratory changes without forgetting earlier conditions, as shown for 3D human pose estimation [32], and cross-checking geometric state against instrument-level feedback as in closed-loop quality control [15], are natural extensions that would relax these constraints.

C. GENERALIZATION OF THE LLM COMPONENTS BEYOND THE TRAINING DISTRIBUTION

Both LLM components are fine-tuned on combinatorially constructed datasets, and although the evaluation sets include unseen wording, sentence forms, and object types, every evaluated command remains within the six-operator XDL schema and the 12-grid workspace abstraction. Three regimes therefore remain uncharacterized. Chemically ambiguous instructions, such as underspecified volumes or vessels, yield protocols whose violated preconditions the XDL Validator rejects with the offending step, but a systematic benchmark across ambiguity classes is left to future work. Adversarial or out-of-schema prompts are contained by the schema-restricted output space, since operators outside the XDL vocabulary fail syntactic validation, but this containment has not been stress-tested against deliberately crafted inputs. Substantially novel apparatus such as separatory funnels or condensers would require extending both the operator schema and the end-effector inventory. Finally, the 1B-parameter backbone was chosen for its inference latency, and the minimum model capacity required for reliable protocol generation and action

reasoning remains open.

D. ERROR PROPAGATION AND SYSTEM-LEVEL ROBUSTNESS

The hierarchical pipeline raises the concern that errors in early modules propagate downstream. The framework contains such propagation through three mechanisms. First, the XDL Validator rejects protocols that violate the 25 conditions of Table 11 before any physical planning begins. Second, the stage-wise World State refresh before and after every procedure (Algorithm 1) re-grounds planning in observed geometry, preventing pose errors from accumulating across stages. Third, bounded retry with re-initialization at both the task- and motion-planning levels absorbs transient planning failures, and unrecoverable failures halt the workflow with a diagnostic message rather than proceeding on an invalid plan.

This containment works only within what each mechanism checks. The XDL Validator rejected 30 of the 34 incorrect protocols that the generator actually produced (Section IV-B2). The 4 that were not rejected satisfy every condition but select an operation the command did not ask for. Such protocols are physically feasible, so they also pass planning and execution, and a procedure the user did not intend is carried out. Because the operator accuracy of the generator falls as commands move away from the training distribution (Table 1), this path opens more often for commands phrased in unseen ways. The decisions of the Action Reasoner are filtered for physical feasibility by the TAMP Module. If the object cannot be placed in the designated grid, no plan that satisfies the constraints is produced, so an infeasible decision is not executed.

In Table 6, the success rate in the perception state was at the same level as in the ground-truth state, and the only incompleteness that arose from perception was the one trial that stopped because the marker was not found. No case was therefore observed in which pose estimation error propagated into planning and execution. This observation concerns the uncertainties the simulation includes, namely the initial configuration, the pose estimation error, and rigid-body contact. The execution factors the simulation does not reproduce are discussed in Section V-A.

Two design choices deserve a precise reading. The 12-grid discretization keeps the output space of the Action Reasoner small without limiting placement precision, because cuTAMP resolves exact coordinates in continuous space, and the target grid is accordingly scored by whether it lies within the zone that the ground-truth rule admits rather than against a single cell (Table 4). The bounded latency of the TAMP Module is a per-round bound, since the batched solver executes a fixed number of optimization steps over a fixed particle budget. The solve times of Table 3 and Fig. 7 gather by round and already include the restarts within the budget.

VI. CONCLUSION

This paper addressed three coupled limitations of chemistry laboratory automation, namely pipelines that require man-

ual redesign for new apparatus or procedures, unreliable constrained motion planning with unpredictable termination time on low-redundancy manipulators, and LLM-based planners confined to protocol generation. The proposed LLM-Guided Tool-Aware TAMP framework unifies LLM-based context-aware decision-making with batched joint differentiable optimization-based planning within a single end-to-end pipeline.

The LLM Module jointly addresses the first and third limitations. Because it converts natural-language commands into validated XDL protocols, the user can instruct each new procedure in natural language rather than writing its protocol by hand. By further inferring the appropriate end-effector and obstacle rearrangement strategy from the current World State and procedural context, it extends the role of the LLM beyond protocol generation to execution-context determination. It reaches 100.0% tool selection accuracy over 1,600 scenarios, including object types absent from the fine-tuning data. In transitions where the next procedure requires a particular station, context-aware rearrangement reduces the mean number of manipulations by 37.5%, and on the Transfer → Stir sequence it reaches a sequence completion rate of 96.67% against 50.0% for random rearrangement. These results hold within the evaluated six-operator schema and 12-grid workspace. The TAMP Module addresses the second limitation by replacing sequential parameter-binding with batched joint differentiable optimization, improving the planning success rate from 73.3% to 100.0% on the Transfer task where orientation constraints are most restrictive, while the time of one optimization round falls within a narrow range for each task, as time-sensitive chemical workflows require. Together, these capabilities enable the integrated framework to achieve a 98.9% end-to-end success rate with the state estimated by the Perception Module on long-horizon chemical workflows initiated from a single natural language command, the same level as the 97.8% obtained with the true state of the simulator. On the physical FR5, adaptive pouring reached a mean absolute error of 0.19–0.34 g for water and an oligosaccharide syrup of about 3,000 cP and 0.36–0.73 g for powdered sugar over target masses of 10–50 g, and the Transfer procedure planned from the perceived positions was completed in all nine runs.

Future work will address the limitations discussed in Section V. Physical validation will be extended to whole workflows that include tool changing and Stir, and a check that compares the intent of the command with the generated operator will reduce the operation-selection errors that the XDL Validator does not reject. The supported XDL operators and skill library will also be expanded to cover more complex chemical procedures such as titration, filtration, and centrifugation, and the LLM components will be benchmarked across model scales and out-of-distribution instruction regimes.

TABLE 9. Simulation environment and generation of the initial configurations.

Item	Setting
Simulator	NVIDIA Isaac Sim 6.0.0.1
Compute	One workstation (AMD Ryzen 7 9800X3D, NVIDIA RTX 5080 16 GB, 30 GB RAM)
Physics integration	Step of 1/60 s, one substep, TGS solver with 8 position and 1 velocity iterations per object
Trajectory execution	Issued against the same 60 Hz step, joint velocity limit of 0.625 rad/s
Joint drives	Joints 1–3: stiffness 1008.44 N-m/deg and damping 100 N-m-s/deg. Joints 4–6: stiffness 379.82 N-m/deg and damping 1.5 N-m-s/deg
Bench	Twelve cells spaced 0.30 m apart. The cell centers are twelve of the combinations of $x \in \{-0.15, 0.15, 0.45\}$ m with $y \in \{-0.6, -0.3, 0, 0.3, 0.6\}$ m, the row at $x = -0.15$ m using only $y = \pm 0.6$ m
Object position (x, y)	Drawn uniformly within ± 5 cm of the cell center
Object yaw	Drawn uniformly within $\pm 180^\circ$
Six arm joints	Drawn uniformly within $\pm 10^\circ$ of the home configuration
Configuration seeds	0–29 per task. Every planner evaluated in Section IV-C receives the same configurations. The random seed of each planner is fixed per initial configuration and changed at each restart

TABLE 10. Settings of the two planners used in the planner comparison. Both planners share the same initial configurations, object sizes, symbolic domain, goal conditions, upright bound $\theta_{\max} = 5^\circ$, and 60 s time budget, and an attempt that ends without a plan is restarted while budget remains.

Item	Setting
<i>Proposed planning module</i>	
Implementation	cuTAMP 0.0.1, cuRobo 0.7.7
Compute	GPU
Collision checking	50 spheres of radius 5 mm per object, zero margin
Search	One plan skeleton, 1,024 particles, 1,000 optimization steps per round
Optimizer	Adam, learning rate 7×10^{-3} (2.226×10^{-2} for the arm configuration)
Cost weights	Kinematic position 1, rotation 5, goal support 2, trajectory length 10^{-3} , all other terms 1
Acceptance	Collision costs below 0.001, placement containment within 0.001 and support within 0.01, joint limits and self-collision met exactly, kinematic error within 0.005 m and 0.05 rad
<i>Baseline (PDDLStream)</i>	
Implementation	Public implementation (October 2023), FastDownward
Compute	CPU, single thread
Collision checking	Object box shapes, zero margin
Search	Adaptive algorithm, unit costs, at most 10,000 iterations
Streams	Grasp, placement pose (200 attempts), inverse kinematics (10 attempts), motion planning

APPENDIX A EXPERIMENTAL SETTINGS

Table 9 summarizes the simulation environment and the generation of the initial configurations, and Table 10 the settings of the two planners compared in Section IV-C.

Planning success and execution success are defined as follows for all three tasks of Section IV-A2. A trial counts as a planning success when a collision-free trajectory that satisfies the upright bound $\theta_{\max}$ is returned within the time budget. It counts as an execution success when all of the following hold

after a settling period of 2 s.

- The plan has been carried out in full.
- The target vessel stands within 30° of upright.
- The target rests inside the goal region. The half-width of the goal region is 50 mm for the vessels, 75 mm for the box, and 15 mm for the stir bar.
- The stir bar additionally lies inside the mouth of the flask, that is, within the mouth radius of 32 mm from the vessel center and below the mouth.
- For Transfer, the pouring motion was performed.

APPENDIX B DATA SEPARATION AND VALIDATOR CONDITIONS

XDL Generator. The training, validation, and evaluation commands are separated at five levels: sentence form, wording, object, composition, and numerical value. Sentence form and wording are separated through the sentence templates. Of 104 sentence templates, 62 are used in training, 12 in a development set used to select the checkpoint, and 30 in evaluation, the latter divided into 18 with unseen wording and 12 with unseen sentence forms. One of the four names of every vessel type appears only in evaluation. The numerical values of each range are split into disjoint pools, 20 and 20 values for volume, 11 and 10 for temperature, and 15 and 15 for time. Three ordered operator pairs (CleanVessel→Add, HeatChill→Transfer, Stir→Move) are withheld from training. Each evaluation set differs from the training data at only one of these levels, and a matched control that follows the training distribution and a set that differs at all five levels are added. The generated data files were checked exhaustively to confirm that no command occurs in both the training and the evaluation data, and a validation set of 500 commands is used to monitor training.

Action Reasoner. Training and evaluation scenarios are separated along three axes. For objects, each of the four types (beaker, flask, box, bottle) has four instance names, the fourth of which (for example, beaker_C) is reserved for evaluation, and the test tube and tray types are not used in training. For spatial relations, 8 of the ordered pairs of grids holding the target object and the obstacle (G3–G8, G4–G9, G5–G10, G6–G11, and their reverses) are withheld from training, while every grid still appears in training both as a target position and as an obstacle position. For task combinations, 6 pairs of current and next operators (Add→Stir, Stir→Transfer, HeatChill→Move, Transfer→HeatChill, CleanVessel→Add, Move→CleanVessel) are withheld from training, while every operator still appears in training both as the current and as the next procedure.

The 25 conditions verified by the three validation checks of the XDL Validator are listed in Table 11.

APPENDIX C ADDITIONAL ILLUSTRATIONS AND RESULTS

This appendix collects illustrations and supporting results referred to in the main text.

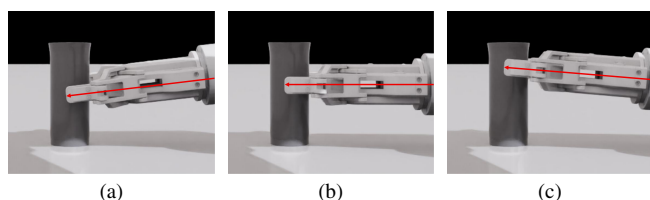
Marker search with the wrist camera. Finally, the marker search with the wrist camera when the fixed-camera obser-

TABLE 11. The 25 conditions verified by the three validation checks.

Check	Conditions verified
Syntactic (10)	unparsable document, wrong root element, procedure with no steps, operator outside the vocabulary, violation of the case convention, missing required attribute, undeclared attribute, unparsable quantity literal, wrong unit, malformed boolean value
Object existence (6)	identifier absent from the object list, identifier absent from the current workspace, reagent absent from the reagent list, identifier used in the wrong argument role, quantity absent from the command, object absent from the command
Physical feasibility (9)	parameter outside the operating range, transfer whose source and destination coincide, operation on an empty vessel, transfer exceeding the volume remaining, vessel capacity exceeded, an enumerated reagent pair brought together in one vessel, reuse of an uncleaned vessel, device or placement already occupied, end-effector not held

TABLE 12. LLM-based tool assignment accuracy by task and target object type, over the 1,600 evaluation scenarios of Section IV-D1. Object types marked with an asterisk are absent from the fine-tuning data.

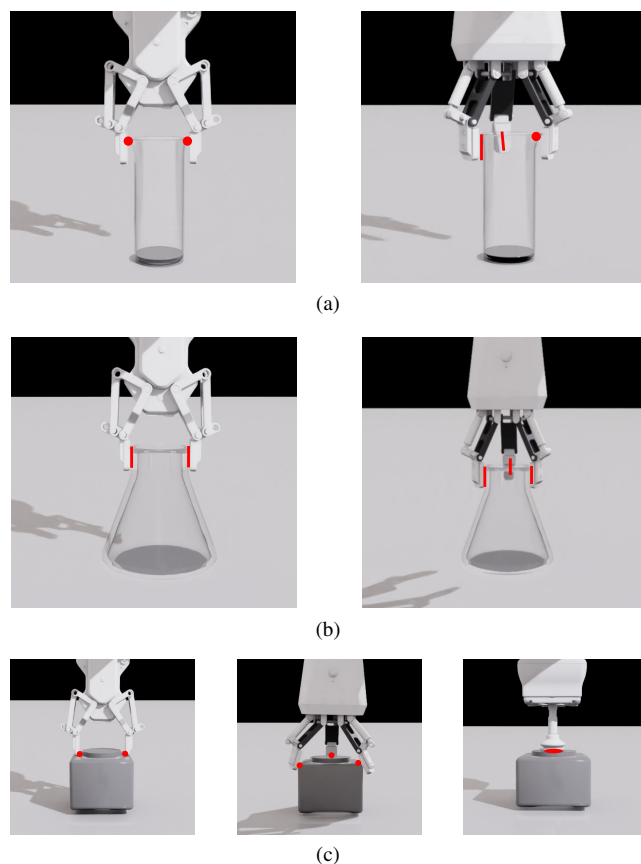
Task	Selected Tool	Target Object	<i>n</i>	Accuracy (%)
Add	3-finger gripper	Beaker, Flask, Tube*	289	100.0
CleanVessel	3-finger gripper	Beaker, Flask, Tube*	306	100.0
HeatChill	3-finger gripper	Beaker, Flask, Tube*	277	100.0
Stir	3-finger gripper	Beaker, Flask, Tube*	239	100.0
Move	3-finger gripper	Beaker, Flask, Tube*	159	100.0
	Suction gripper	Bottle, Box, Tray*	165	100.0
Transfer	2-finger gripper	Beaker, Flask, Tube*	165	100.0
All tasks			1,600	100.0

**FIGURE 12.** Discrete yaw candidates for the SidePick operator, illustrating how gripper tilt varies across the sampled configurations. (a) $\gamma = -\pi/2.5$, (b) $\gamma = \pi/2$ (or $-\pi/2$), (c) $\gamma = \pi/2.5$.

variations are insufficient was checked on the physical robot. In the 30 g water run, the Perception Module recognized both beakers with the two fixed cameras, but the tag of the receiving beaker appeared small in the images and fell short of the measurement quality criterion. The arm therefore carried the wrist camera through 36 predefined viewpoints above the workspace in turn, dwelling about 1.5 s at each to search for the marker (Fig. 11). At the 21st viewpoint the wrist camera read the tag of the receiving beaker larger and sharper than the fixed cameras, and this measurement updated its position. The Transfer of this run was executed with the trajectory planned from the two beaker positions confirmed in this way.

REFERENCES

- [1] M. Seifrid et al., "Autonomous chemical experiments: Challenges and perspectives on establishing a self-driving lab," *Accounts Chem. Res.*, vol. 55, no. 17, pp. 2454–2466, 2022.
- [2] M. Abolhasani and E. Kumacheva, "The rise of self-driving labs in chemical and materials sciences," *Nature Synth.*, vol. 2, no. 6, pp. 483–492, 2023.
- [3] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 4, no. 1, pp. 265–293, 2021.
- [4] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Learning compositional models of robot skills for task and motion planning," *Int. J. Robot. Res.*, vol. 40, no. 6–7, pp. 866–894, 2021.
- [5] N. Yoshikawa et al., "Chemistry lab automation via constrained task and motion planning," *arXiv:2212.09672*, 2022.

**FIGURE 13.** Motivation for multi-gripper tool selection. The 2-finger gripper provides insufficient contact area on the rim of cylindrical vessels, whereas the 3-finger gripper achieves stable top-down grasping. For large flat-surface objects exceeding the graspable width of finger grippers, the suction gripper is required. (a) Beaker, (b) Flask, (c) Large flat-surface object.

- [6] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "PDDLStream: Inte-

- grating symbolic planners and blackbox samplers via optimistic adaptive planning,” in *Proc. Int. Conf. Automated Planning Scheduling*, vol. 30, 2020, pp. 440–448.
- [7] Z. Kingston, M. Moll, and L. E. Kavraki, “Exploring implicit spaces for constrained sampling-based planning,” *Int. J. Robot. Res.*, vol. 38, no. 10–11, pp. 1151–1178, 2019.
- [8] W. Shen, C. Garrett, N. Kumar, A. Goyal, T. Hermans, T. Lozano-Perez, and F. Ramos, “Differentiable GPU-parallelized task and motion planning,” in *Proc. Robot., Sci. Syst. (RSS)*, 2025.
- [9] B. Burger et al., “A mobile robotic chemist,” *Nature*, vol. 583, no. 7815, pp. 237–241, 2020.
- [10] H. Fakhroldeen, G. Pizzuto, J. Glowacki, and A. I. Cooper, “ARChemist: Autonomous robotic chemistry system architecture,” in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2022, pp. 6013–6019.
- [11] A. M. Lunt et al., “Modular, multi-robot integration of laboratories: An autonomous workflow for solid-state chemistry,” *Chem. Sci.*, vol. 15, no. 7, pp. 2456–2463, 2024.
- [12] M. Kennedy, K. Schmeckpeper, D. Thakur, C. Jiang, V. Kumar, and K. Daniilidis, “Autonomous precision pouring from unknown containers,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2317–2324, 2019.
- [13] Y. Huang, J. Wilches, and Y. Sun, “Robot gaining accurate pouring skills through self-supervised learning and generalization,” *Robot. Auton. Syst.*, vol. 136, 2021, Art. no. 103692.
- [14] N. Yoshikawa, G. D. Akkoc, S. Pablo-García, Y. Cao, H. Hao, and A. Aspuru-Guzik, “Does one need to polish electrodes in an eight pattern? Automation provides the answer,” *Digit. Discovery*, vol. 4, no. 2, pp. 326–330, 2025.
- [15] S. U. Khan, V. K. Møller, R. J. N. Frandsen, and M. Mansourvar, “Real-time AI-driven quality control for laboratory automation: A novel computer vision solution for the Opentrons OT-2 liquid handling robot,” *Appl. Intell.*, vol. 55, no. 7, 2025, Art. no. 524.
- [16] R. I. C. Muchacho, R. Laha, L. F. C. Figueredo, and S. Haddadin, “A solution to slosh-free robot trajectory optimization,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2022, pp. 223–230.
- [17] Z. Kingston, M. Moll, and L. E. Kavraki, “Sampling-based methods for motion planning with constraints,” *Annu. Rev. Control Robot. Auton. Syst.*, vol. 1, no. 1, pp. 159–185, 2018.
- [18] D. Coleman, I. Sucan, S. Chitta, and N. Correll, “Reducing the barrier to entry of complex robotic software: A MoveIt! case study,” *arXiv:1404.3785*, 2014.
- [19] Y. Li, L. Ke, and C. Zhang, “Dynamic obstacle avoidance and grasping planning for mobile robotic arm in complex environment based on improved TD3,” *Appl. Intell.*, vol. 55, no. 11, 2025, Art. no. 776.
- [20] K. Kawaharazuka, J. Oh, J. Yamada, I. Posner, and Y. Zhu, “Vision-language-action models for robotics: A review towards real-world applications,” *IEEE Access*, vol. 13, pp. 162467–162504, 2025.
- [21] Q. Peng, C. Bai, G. Zhang, B. Xu, X. Liu, X. Zheng, C. Chen, and C. Lu, “NavigScene: Bridging local perception and global navigation for beyond-visual-range autonomous driving,” in *Proc. 33rd ACM Int. Conf. Multimedia (MM)*, 2025.
- [22] S. H. M. Mehr, M. Craven, A. I. Leonov, G. Keenan, and L. Cronin, “A universal system for digitization and automatic execution of the chemical synthesis literature,” *Science*, vol. 370, no. 6512, pp. 101–108, 2020.
- [23] N. Yoshikawa et al., “Large language models for chemistry robotics,” *Auton. Robots*, vol. 47, no. 8, pp. 1057–1086, 2023.
- [24] K. Darvish et al., “ORGANA: A robotic assistant for automated chemistry experimentation and characterization,” *Matter*, vol. 8, no. 2, 2025.
- [25] A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White, and P. Schwaller, “ChemCrow: Augmenting large-language models with chemistry tools,” *arXiv:2304.05376*, 2023.
- [26] D. A. Boiko, R. MacKnight, B. Kline, and G. Gomes, “Autonomous chemical research with large language models,” *Nature*, vol. 624, no. 7992, pp. 570–578, 2023.
- [27] S. H. Vemprala, R. Bonatti, A. Buckner, and A. Kapoor, “ChatGPT for robotics: Design principles and model abilities,” *IEEE Access*, vol. 12, pp. 55682–55696, 2024.
- [28] Y. Qu, X. Hu, F. Chen, and Z. Wei, “TRTP: A three-stage robust task planning framework for open worlds via visual-language models and digital twin simulation,” *Appl. Intell.*, vol. 56, no. 4, 2026, Art. no. 107.
- [29] J. Wang and E. Olson, “AprilTag 2: Efficient and robust fiducial detection,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2016, pp. 4193–4198.
- [30] H. Sarmadi, R. Muñoz-Salinas, M. A. Berbís, and R. Medina-Carnicer, “Simultaneous multi-view camera pose estimation and object tracking with squared planar markers,” *IEEE Access*, vol. 7, pp. 22927–22940, 2019.
- [31] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
- [32] Q. Peng, H. Xue, P. Wang, and C. Chen, “Lifelong domain adaptive 3D human pose estimation,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 40, no. 10, 2026, pp. 8358–8366.



BOHYEONG PAK received the B.S. degree from the Department of Mechanical Engineering, Hanbat National University, in 2025. He is currently pursuing the M.S. degree with the Department of Mechanical Engineering, Kyung Hee University. His current research interests include collision avoidance, manipulation, and vision language model planning.



HYUNHO CHO received the B.S. degree from the Department of Mechanical Engineering, Hanbat National University, in 2025. He is currently pursuing the M.S. degree with the Department of Mechanical Engineering, Kyung Hee University. His current research interests include optimal control and reinforcement learning.



SANGHYUN KIM (Member, IEEE) received the B.S. degree in mechanical engineering and the Ph.D. degree in intelligent systems from Seoul National University, South Korea, in 2012 and 2020, respectively. He was a Postdoctoral Researcher with The University of Edinburgh, U.K., in 2020, and a Senior Researcher with the Korea Institute of Machinery and Materials, South Korea, from 2020 to 2023. He is currently an Assistant Professor in mechanical engineering with Kyung Hee University. His main research interest includes the optimal control of mobile manipulators.

...