ManiSkillFormer: Demonstration-Free Compositional Manipulation via Geometric Contracts and Agentic Skill Graph

Peiqi Yu, Mosam Dabhi, Shangtao Li, Bowei Li, Laszlo Jeni, and Changliu Liu
Carnegie Mellon University, Pittsburgh, PA 15213, USA
Email: {peiqiy, mdabhi, cliu6, shangtal, boweili, laszloaj, cliu6}@andrew.cmu.edu

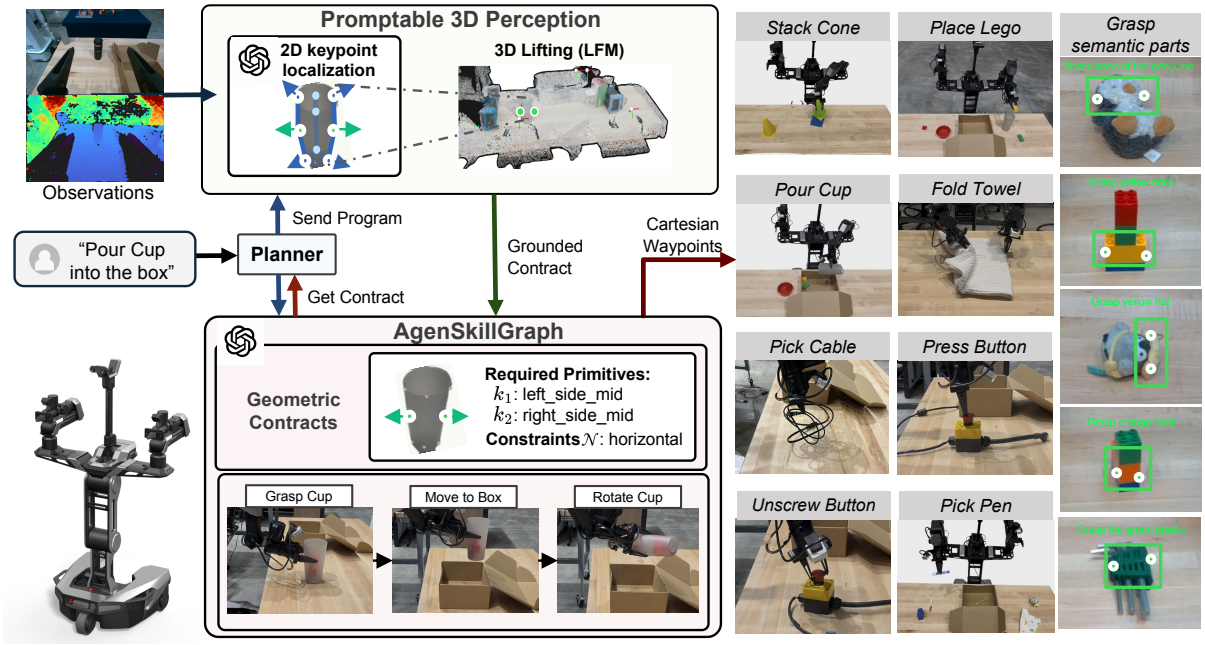

Fig. 1: **ManiSkillFormer online stage overview.** Given a language instruction, the Planner maps the task to structured skill-object pairs and retrieves the corresponding geometric contracts from AgenSkillGraph. A promptable 3D perception module then grounds the requested semantic 3D primitives from calibrated multi-view observations. The grounded execution packet parameterizes agent-generated motion templates in the SkillGraph to generate Cartesian waypoints. Right side: representative demonstration-free manipulation across different task types.

***Abstract*— Adapting robotic manipulation to new objects and tasks often requires additional demonstrations, policy fine-tuning, or manual engineering. Reusable manipulation skills can reduce this effort, but connecting their execution requirements to scene-specific geometry remains challenging. We present ManiSkillFormer, a framework for demonstration-free and compositional manipulation that connects perception and action through explicit geometric contracts. Building on reusable skill schemas, LLM agents generate contracts specifying the geometry primitives required by each skill, together with corresponding motion templates for semantic objects and task contexts. These contracts guide a perception module to ground task-relevant 3D geometry from observations, which is then used to instantiate reusable motion templates in a skill library. We evaluate ManiSkillFormer on a dual-arm robot across demonstration-free pick-and-place with 30 instances from 8 object categories, functional manipulation including unscrewing, pouring, pressing, and folding, and three long-horizon tasks. ManiSkillFormer achieves an average success rate of 88.97% for pick-and-place, 75.00% for functional manipulation, and completion rates of 50–80% across the long-horizon tasks, outperforming the evaluated baselines and two ablated pipelines. These results demonstrate the potential of explicit geometric contracts to support skill reuse and composition across objects and tasks without per-object policy fine-tuning or additional robot demonstrations.**

I. INTRODUCTION

Robotic manipulation in unstructured environments requires behaviors that generalize across changing objects and compose into long-horizon tasks. Recent vision-language-action (VLA) models address this challenge by learning end-to-end policies that map observations and instructions directly to actions [1], [2]. Although these policies achieve broad task coverage, they require substantial embodied data and computational resources. More importantly, because perception, task reasoning, and control are entangled within a single learned mapping, it is often difficult to interprete its action selection, identify the source of a failure, or anticipate how the policy will behave in unfamiliar situations.

Hierarchical and modular approaches address this limitation by separating high-level task reasoning from reusable

robot skills or programs [3]–[6]. This decomposition makes the system more inspectable, but also brings a problem: how do these separated modules communicate with each other in an interpretable, efficient, yet generalizable way?

Let's look at how humans do this. Consider how a human expert might teach a novice to program a robot. Rather than specifying every Cartesian waypoint, the expert provides mostly semantic guidance: where an object should be grasped and which meaningful intermediate waypoints the robot should pass through. For example, when instructed to "grasp the cup to pour," the novice may identify two opposing points on the cup's mid-body and select a horizontal approach direction compatible with the subsequent rotation. The corresponding motion can also be described semantically: move above the cup, approach along the selected direction, and close the gripper, without bringing in exact coordinates. Once these references are located in the current scene, the semantic description can be instantiated as numerical robot commands. This process suggests a natural decomposition: *the geometric requirements and motion structure of a skill can be represented in semantic space, while only their scene-specific realization and final execution need to be instantiated numerically*.

Inspired by this decomposition, we propose ManiSkill-Former, a framework for demonstration-free and compositional robotic manipulation that connects a promptable 3D perception module with a reusable skill library, AgenSkillGraph, through *task-conditioned semantic geometric contracts*. Each skill in AgenSkillGraph pairs a contract specifying what keypoints perception must provide with a semantic skill template specifying how the grounded geometry should become object-relative waypoints. Before deployment, schema-guided agents instantiate contracts and skill templates. At runtime, a planner retrieves and composes the required skills, the promptable 3D perception module then grounds their requested primitives in the observed scene according to the contract, and the generated skill templates instantiate them as numerical robot waypoints. This design preserves reusable skill requirements while adapting their numerical realization to each scene and object.

Our main contributions are:

1) *Semantic geometric contracts.* We introduce an interface letting reusable skills explicitly request the geometry they need from perception, satisfying the needs from both the task side and the skill execution side.
2) *A contract-grounded AgenSkillGraph.* We build AgenSkillGraph with contract-grounded atomic skills and dependency-aware meta skills, semi-autonomously instantiated before deployment by schema-guided agents.
3) *Real-robot validation.* On the Galaxea R1-Lite, we evaluate generalization to different object instances, functional manipulation, and long-horizon composition.

II. RELATED WORK

A. Reusable Skills and Structured Manipulation

Skill-based manipulation addresses two complementary concerns: organizing reusable behaviors and coordinating their execution. MOSAIC [5] and NeSyPack [6] support modular execution through skill libraries and hierarchical abstractions. Beyond modularity, STAP accounts for geometric dependencies between successive skills using learned feasibility estimates [7]. Building on NeSyPack's SkillGraph, we compose basic actions (atomic skills) into reusable sequences (meta skills), providing a compact, semantically meaningful vocabulary for LLM planning.

B. Task-Relevant Geometric Perception

Recent VLMs support language-conditioned 2D point prediction [8], [9], making them suitable for localizing the interaction points described by geometric contracts. Using these predictions for manipulation requires recovering their 3D geometry, but direct back-projection relies on depth measurements that may be incomplete or noisy. Lifting foundation models complement 2D localization by inferring 3D structure from landmarks across object categories [10]. Thus, we combine VLM 2D landmark prediction with a SoTA 3D lifting model to reduce reliance on incomplete or noisy depth measurements.

C. Perception–Action Interfaces

Previous interfaces connect perception to action through constraints or learned policies. Explicit approaches guide motion generation using value maps, as in VoxPoser [11], or geometric constraints, as in ReKep, CoPa, and GeoManip [12]–[14]. MOKA and OmniManip express interactions through visual marks and object-centric primitives [15], [16]. Learned approaches instead condition execution on keypoints or spatial primitives [17]–[19]. Our contracts provide an explicit interface specifying the task-dependent interaction geometry required by reusable motion templates.

D. LLM-Based and Agentic Manipulation

LLM-based frameworks organize robot behavior at different levels of abstraction. At the skill level, SayCan selects actions using learned affordances [3], while RoboMatrix combines task decomposition with execution checking over learned meta-skills [20]. At the program level, Code as Policies generates executable logic over perception and control APIs [4]. Our agents focus on skill adaptation, tailoring human-defined skills to different objects and task contexts by jointly specifying what to perceive and how to move.

III. PROBLEM FORMULATION

We consider tabletop lightweight manipulation: a dual-arm manipulator with parallel-jaw grippers acting on rigid or mildly deformable objects that rest within reach on a planar work surface. The action vocabulary is constrained: manipulation verbs (skills) are drawn from a fixed set $\mathcal{V}$ of tabletop manipulation verbs, chosen to be representative of common tabletop tasks. Object nouns are open-vocabulary.

To formalize the problem, we introduce three spaces:

- **Semantic space** $\mathcal{L}$ contains user instructions and semantic relationship between objects and skills, which is represented and reasoned in the language space.

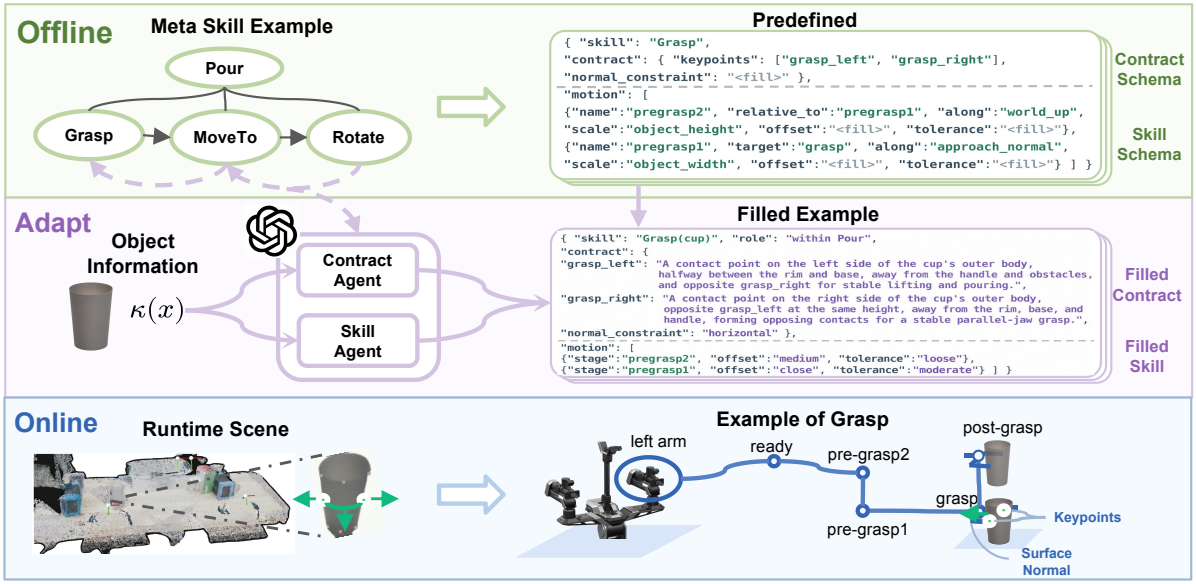

Fig. 2: Overview of the Agentic SkillGraph pipeline. Offline, human experts define reusable skill structurs and contract schemas. During adaptation, two LLM agents tailor these schemas to each semantic skill–object pair and its task context, generating geometric contracts and semantic motion templates. Online, the perception module grounds the requested geometry in the observed scene, which parameterizes the motion templates for robot execution. Green denotes human-defined fields, and purple denotes agent-generated fields.

- **Observation space** $\mathcal{O}$ contains raw robot observation and its scene-specific geometric grounding such as 3D keypoints and surface normals.
- **Numerical space** $\mathcal{N}$ contains robot execution parameters, including waypoints in Cartesian space and joint space τ_i.

The robot receives an instruction $l \in \mathcal{L}$ and an initial observation $o_1 \in \mathcal{O}$. We assume that l specifies an ordered sequence of skill calls, each involving at most one in-hand object and one target object:

$$l = \big((s_1, \mathbf{X}_1), \ldots, (s_H, \mathbf{X}_H)\big) \in \mathcal{L}, \quad \mathbf{X}_i = (x_i^{\mathrm{h}}, x_i^{\mathrm{t}}) \quad (1)$$

where $s_i \in \mathcal{V} \subseteq \mathcal{L}$ is a symbolic skill call, and $x_i^{\mathrm{h}}, x_i^{\mathrm{t}} \in \mathcal{X} \cup \{\varnothing\}$ are symbolic arguments specifying the in-hand and target objects respectively, with $\mathcal{X} \subseteq \mathcal{L}$.

Executing each call requires grounding its symbolic object arguments in the observation space and its symbolic skill in the robot space. We denote the grounded object arguments by $\hat{\mathbf{X}}_i \in \mathcal{O}$ and the grounded skill by $\tau_i = \hat{s}_i(\theta_i, \hat{\mathbf{X}}_i)$, where $\theta_i \in \mathcal{N}$ denotes its skill parameters, $\hat{s}_i$ denotes its skill policy, and τ_i denotes the final trajectory segment in the robot joint space.

For example, consider a call that picks up a mug: $(s_i, \mathbf{X}_i) = (\texttt{Pick}, (\texttt{mug}, \varnothing)) \in \mathcal{L}$. The symbolic argument `mug` is first grounded to the observed mug $\hat{\mathbf{X}}_i \in \mathcal{O}$. Based on its observed geometry, θ_i may specify grasp pose, approach direction, and lift distance in the numerical space as grounded execution parameters. Based on θ_i and $\mathbf{X}_i$, The skill policy $\hat{s}_i$ then generates the trajectory segment $\tau_i \in \mathcal{N}$, which specifies the sequence of robot motions used to execute the call.

In general, given (u, o_1), the objective is to ground each symbolic call $(a_i, \mathbf{X}_i)$ and produce an executable trajectory sequence $\tau = (\tau_1, \ldots, \tau_H)$ to complete the instruction l.

Algorithm 1 Online execution of ManiSkillFormer

Require: instruction l, observation o_0, AgenSkillGraph $\mathcal{G}$, promptable 3D perception module, robot controller $\mathcal{C}_r$
Ensure: execution success or failure

$\Phi \leftarrow \mathrm{InitState}(o)$
$p \leftarrow \mathrm{Plan}(l; \mathcal{G})$ ▷ via $\mathcal{D}$
$c_p \leftarrow \mathrm{RetrieveContract}(p; \mathcal{G})$
$\boldsymbol{\theta} \leftarrow \mathit{Perception}(o, c_p)$ ▷ ground requested primitives
$\hat{p} \leftarrow ((a_i, x_i, \theta_i))_{i=1}^{N}$
for $i = 1$ to N **do**
  $(a_i, x_i, \theta_i) \leftarrow \hat{p}[i]$
  if $\neg \mathrm{Pre}_{a_i}(x_i, \Phi)$ **then return False**
  end if
  $\mu_i \leftarrow \mathrm{RetrieveTemplate}(a_i, \kappa(x_i); \mathcal{G})$
  $\tau_i \leftarrow \mu_i(\theta_i)$
  $\mathcal{C}_r(\tau_i)$
  $\Phi \leftarrow \mathrm{UpdateState}(\Phi, a_i, x_i)$
  if $\neg \mathrm{Post}_{a_i}(x_i, \Phi)$ **then return False**
  end if
end for
return True

IV. METHOD

A. System Overview

As shown in Figure 1, method has four components: a rule-based *planner* that parses instructions into skill–object calls; a promptable 3D perception module that grounds a

skill's geometric requirements on the observed object; a skill library *AgenSkillGraph* $\mathcal{G}$ storing reusable skill definitions; and an embodiment-dependent *controller*. The promptable 3D perception module and AgenSkillGraph are connected through task-conditioned geometric contracts. The framework operates in three stages (Figure 2): offline schema defining; semi-online skill adaptation; and online execution.

Offline Stage—Schema Defining. In the offline stage, we need to prepare an AgenSkillGraph for the specific task.

To prepare a AgenSkillGraph $\mathcal{G}$, we (as human experts) first define semantic atomic skills $\mathcal{A}$, meta skills $\mathcal{M}$, and decomposition rules $\mathcal{D}$, where $\mathcal{D}$ specify how atomic skills are organized to realize each meta-skill. Then we should define a contract schema Σ_a^c and a skill schema Σ_a for each atomic skill $a \in \mathcal{A}$. The contract schema Σ_a^c specifies what primitive labels a skill might require for perception module to ground, and leave blank for agent in semi-online stage to fill in the semantic descriptions for these primitives and approach normal constraints (see Sec. IV-C for details). The skill schema Σ_a fixes the ordered motion stage of a in semantic space (see Sec. IV-E for details).

These design are in semantic space $\mathcal{L}$, maintaining human knowledge in a easily interpretable and configurable way but are irrelevant to specific scene, reducing manual scene-by-scene skill specification while constraining subsequent agent implementation to remain interpretable and consistent.

Semi-online—Plan and Skill Adaptation. Given instruction l, planner first plans the skill-object sequence:

$$p = \mathrm{Plan}(l; \mathcal{G}) = ((s_1, x_1), \ldots, (s_k, x_k)) \in \mathcal{L} \qquad (2)$$

Each skill s_i is either atomic or meta. For an atomic skill, two LLM agents generate the contract and skill template for the skill-object pair. A meta skill is first expanded into its constituent atomic skills a_i, after which the agents generate a contract and skill template for each atomic skill, conditioned on its role within the meta skill (see Sec. IV-C for details):

$$\begin{aligned} c_{a_i,\kappa(x_i)} &= \mathrm{FillContract}(\Sigma_{a_i}^c, \Sigma_{a_i}, \kappa(x_i)) \in \mathcal{L}, \\ \mu_{a_i,\kappa(x_i)} &= \mathrm{FillSkill}(\Sigma_{a_i}, c_{a_i,\kappa(x_i)}, \kappa(x_i)) \in \mathcal{L} \end{aligned} \qquad (3)$$

x_i is the semantic object name, like "grey toy"; $\kappa(x_i)$ is the object category that can be specified by human, including `rigidity`, `size`, and `shape`. If not specified, the agent would generate purely according to semantic object name x_i; $c_{a,\kappa(x)}$ is the filled contract which states what perception must find; $\mu_{a,\kappa(x)}$ is the filled skill template states how the grounded result becomes Cartesian waypoints; both remain free of scene coordinates. This stage yields an atomic skill execution program $p^c = ((a_i, x_i, c_{a_i,\kappa(x_i)}, \mu_{a_i,\kappa(x_i)}))_{i=1}^N$, attach each step with a filled contract and skill template. In this stage, the filled results are also reasoned in semantic space $\mathcal{L}$, utilizing LLM's world knowledge.

Online—Ground and Execute. Given observation o and the filled program p^c from the semi-online stage, ManiSkill-Former performs

$$\hat{p} = \mathrm{Ground}(o, p^c) \in \mathcal{O}, \quad \tau = \mathrm{Execute}(\hat{p}; \mathcal{G}) \in \mathcal{N} \qquad (4)$$

Ground is done by the promptable 3D perception module, replacing each contract in p^c with its grounded realization θ_i, producing $\hat{p} = ((a_i, x_i, \theta_i, \mu_{a_i,\kappa(x_i)}))_{i=1}^N$. Execute instantiates each trajectory segment $\tau_i = \mu_{a_i,\kappa(x)_i}(\theta_i)$, which is then dispatched to controller $\mathcal{C}_r$.

The below sections would go into module details.

B. Planner

Given instruction l, the planner uses a rule-based parser to extract an ordered sequence of skill–object calls by matching each verb phrase to the skill vocabulary $\mathcal{A} \cup \mathcal{M}$ and its adj-noun pair phrases to object arguments, ordered by clause sequence in l. For example, "pick red lego and place into black cup, then pour black cup into brown box" yields $\{l$: `Pick(red lego,`$\varnothing$`)`, `Place(red lego,black cup)`, `Pour(black cup,brown box)`$\}$. The final result is a skill sequence $p = ((a_1, x_1), \ldots, (a_N, x_N))$.

C. Task-Conditioned Geometric Contracts

A geometric contract $c_{a,\kappa(x)}$ specifies what information perception must provide for skill execution.

Semantic Keypoints and Surface Normal Constraints. The contract should contain keypoint request $k_{a,\kappa(x)} = (\ell_a^i, d_{a,\kappa(x)}^i)_{i=1}^N$: N keypoints, each with a semantic label ℓ_a and a free-form description $d_{a,\kappa(x)}$ of where it lies on the object, e.g., *"center point on one well-exposed side of the bottle's main body"*. It also contains a surface normal constraint $n_{a,\kappa(x)}$, which specifies the admissible surface orientation at the interaction region in the robot base frame. The constraint can specify an angular range relative to a world-frame axis or plane, such as 30°–60° relative to the vertical z-axis. The labels `upright` and `horizontal` provide shorthand for alignment with the vertical z-axis and the plane orthogonal to z, respectively, within a specified angular tolerance. If the constraint is `none`, no surface orientation restriction is imposed. At runtime, the 3D perception module jointly grounds each requested keypoint and its local surface normal, selecting candidate locations whose estimated normals lie within the admissible set $\mathcal{N}_{a,\kappa(x)}$. If the constraint is `none`, no orientation class is imposed. This surface normal would decide end-effector approach direction. For a parallel-jaw grasp, the grounded contact points and the surface normal would help determine the 6-DoF contact pose used to instantiate the skill template.

Example. As shown in Figure 2, the contract schema Σ_a^c should explicitly specifies the semantic keypoints name ℓ_a needed for skill execution in advance (e.g., `grasp_left`/`grasp_right` for `Grasp`, shown in green). A contract agent (gpt-5.6-sol [21]) then fills $d_{a,\kappa(x)}$ and $n_{a,\kappa(x)}$ (purple). We assume a parallel-jaw gripper, so `Grasp` requests two opposing keypoints; other grippers would require a different keypoint set.

Consider `Grasp(cup)` within `Pour`$(\mathrm{cup}, \mathrm{box}) =$ `Grasp` $\rightarrow$ `MoveTo` $\rightarrow$ `Rotate`. Taken alone, `Grasp(cup)` admits either class: a top-down grasp on the rim, whose contact normals are `upright`, or a side grasp on the body, whose normals are `horizontal`. The

downstream `Rotate` eliminates the first, since a rim grasp would cause self collision in the subsequent `Rotate` skill. The agent therefore sets $n_{a,\kappa(x)} = \texttt{horizontal}$ and requests two opposing points on the body wall.

D. Contract-Guided Promptable 3D Perception Module

This module maps observations and a contract request to grounded object-centric 3D primitives in Cartesian space.

Input. This module takes an observation and a contract request. The observation is a set of calibrated camera views. Our platform provides two RGB-D and two RGB views, and this module runs on any subset containing depth or at least two views. The contract request contains the keypoint set $\mathcal{K}_{a,\kappa(x)}$ in Section IV-C, and the skill a and object x it belongs to.

Task-conditioned 2D Keypoint Prediction. We use gpt-5.6-sol to predict semantic 2D interaction points from RGB images. For each view, we construct a textual query specifying the skill, target object, keypoint labels and their descriptions $\{(\ell_a^i, d_{a,\kappa(x)}^i)\}_i$ required by the contract. Requested keypoints are queried jointly, and the model returns image-plane coordinates associated with their semantic labels. These labeled predictions are passed to a 3D lifting model described below to obtain the 3D keypoints used to parameterize the motion templates. No task-specific fine-tuning is required in this stage.

3D Lifting and Constraint Checking. Each predicted 2D keypoint is lifted to 3D from the fused observations using the LFM 3D lifting model [10]. For each candidate set, we estimate the local surface normal $\mathbf{n}$ at the interaction region and express it in the robot base frame. We then compute its angle relative to the axis or plane specified by the contract and check whether this angle falls within the prescribed range. Only candidate sets satisfying $\mathbf{n} \in \mathcal{N}_{a,\kappa(x)}$ are retained for motion parameterization. If the constraint is `none`, this check is skipped.

Robot Frame Grounding. Using the calibrated camera intrinsics and extrinsics, we express each reconstructed 3D keypoint in the robot base frame. For robot-mounted cameras, we compute the camera pose at capture time using the robot's forward kinematics and current joint configuration, accounting for changes in camera pose across observations.

E. AgenSkillGraph: Agent-Assisted Skill Construction

AgenSkillGraph stores human-defined skill schemas and the agent-generated parameterized skills.

Skill Schema. Inspired by how experts teach novices, Σ_a encodes human knowledge as an ordered sequence of semantic motion stages. The stages are chosen based on changes in contact mode or relative robot–object pose; for example, `Grasp` follows `pre-grasp2` → `pre-grasp1` → `grasp` → `post-grasp` → `retract` (see the example in Figure 2). In view that how the novice implement these semantic waypoints should adapt to the size of objects it operates on, each stage specifies an *offset distance* (`close`, `medium`, or `far`). The offset is normalized by a *scale reference*, namely a task-relevant object dimension such as its width or height, and is converted into a category-dependent numerical displacement during adaptation (Sec. IV-E). Also, not all waypoints should be strictly reached, so we specify a *reaching tolerance* (`precise`, `moderate`, or `loose`), allowing contact-critical stages to remain precise while intermediate stages serve as flexible guidance.

TABLE I: Atomic Skills and Meta-Skill Decompositions.

Atomic Skills		Meta Skills	
Contact Mode	Atomic Skill	Meta Skill	Skill Decomposition
No contact	`MoveTo`	`Pick`	`Grasp` → `MoveTo`
Establish contact	`Grasp`	`Place`	`MoveTo` → `Release`
Maintain contact	`Rotate`	`Pour`	`Grasp` → `MoveTo` → `Rotate`
Terminate contact	`Release`	`Unscrew`	`Grasp` → `Rotate`
Interaction under contact	`Press`	`Fold`	`Press` → `Grasp` → `MoveTo`

In addition, Σ_a defines $\chi_a = (\mathrm{Pre}_a, \mathrm{Post}_a)$, whose conditions are evaluated by internal utility functions, such as checking whether the gripper is open or the object is in hand. As shown in Algorithm 1, Pre_a must hold before the skill is attempted and Post_a after its execution; failure of either check terminates execution.

Filling a Motion Template. During adaptation, given a robot reachibility constraint, the skill agent (gpt-5.6-sol) assigns each waypoint an *offset-distance* class and a *reaching-tolerance* class. Each offset-distance class maps to a range expressed as a fraction of the stage's scale reference (e.g., `close` is 0.15–$0.35\times$ the reference); each reaching-tolerance class maps to a fixed absolute range, used for execution to decide how far a waypoint may deviate from its nominal pose. These classes are generated and resolved against the object the skill operates on, which is what lets a single $\mu_{a,\kappa}$ generalize across instances of different size and shape. Figure 2 shows an example of the motion templates of $\Sigma_{\texttt{Grasp}}$, what skill schema pre-defined is colored in green, and what filled by agent is colored in purple.

Atomic and Meta Skills. As shown in Table I, we organize the atomic skill set by *contact mode*, making them the minimal units that are reusable yet interpretable in this paper. Meta skills are compositions of existing atomic skills. We present the meta skills used in this paper in Table I. These meta skills were chosen because they span different contact-transition sequences, are reusable across tasks, and together cover the functional manipulation and long-horizon tasks in our experiments. Decomposition rules $\mathcal{D}$ are predefined; this keeps the atomic sequence for a given meta skill consistent and inspectable. Extending $\mathcal{G}$ to additional meta skills requires simply writing a new decomposition rule.

F. Controller and Execution

Online, the perception module grounds the object parameters in θ_i. Each stage's offset-distance range is scaled by the object dimensions, after which an offset and a tracking tolerance are uniformly sampled from their respective ranges. This yields Cartesian waypoints $X_i^{(j)}$ with tracking tolerances $\epsilon_i^{(j)}$. The controller $\mathcal{C}_r$ assigns arms based on reachability. For dual-arm skills, the template fixes inter-arm coordination, so selecting the arm for the first action determines all subsequent arm assignments. Using the waypoints and tolerances, the controller solves inverse kinematics

subject to self-collision avoidance, generates the joint-space trajectory τ_i, and executes it via low-level PD control.

V. EXPERIMENTS

We evaluate ManiSkillFormer by three research questions:

Q1 (Object generalization). Can skills generalize to different object instances without further demonstrations?

Q2 (Functional manipulation). Does skill-specific contract return the right geometry, including when the same object is used by different skills?

Q3 (Long-horizon composition). Can we compose skills for long-horizon tasks with task dependencies reliably?

A. Experimental Setup

Robots and Sensing. Evaluations are conducted on *Galaxea R1-Lite*, a dual-arm mobile manipulator with wrist RGB-D and head cameras and a parallel-jaw gripper.

Baseline and Ablation Study. We compare against two external baselines and two ablations. **MOKA** [15] uses mark-based visual prompting to predict point-based affordances and generate manipulation motions, providing a comparison with a task-conditioned geometric interface. **VLA system** π_0 [2] is fine-tuned with 50 demonstrations per object category and evaluated on pick-and-place. **A1 (no contract)** passes the task description directly to the promptable 3D perception module, which returns the number of keypoints required by the skill without explicit contract-defined semantic requests. **A2 (independent generation)** generates each atomic skill's contract and motion template independently, without conditioning on subsequent skills in the meta skill. All methods use the same object instances, layouts, task instructions, and success criteria.

Task Suites. We organize the evaluations into three tabletop suites: 1) The *instance-transfer suite* evaluates pick-and-place across 8 object groups comprising 30 instances in total, with 17 trials per group. 2) The *functional manipulation suite* evaluates unscrewing, pouring, pressing, and folding, with 17 trials per task, to examine geometric grounding required by different skills. 3) The *long-horizon compositional suite* evaluates three multi-step tasks, with 10 trials per task, to examine skill composition and dependencies between successive actions.

Metric. A trial is successful if the robot achieves the task goal. We report success rates averaged across repetitions.

B. Results

1) Q1 (Object Generalization)

Table II reports single-object pick-and-place performance across 8 object groups with 30 instances in total (136 trials for each method). According to the results, ManiSkillFormer achieves the highest or joint-highest success rate in seven of the eight object groups, supporting the robustness of contract generation and the reuse of generated motion templates in demonstration-free cases without per-instance tuning.

Where contracts help most. ManiSkillFormer outperforms MOKA and π_0 on all eight object groups. Compared with A1 (no contract), ManiSkillFormer achieves the largest

TABLE II: Single-object pick-and-place success rates (%).

Object Group	ManiSkillFormer	MOKA	A1	A2	π_0
Bottle / Cup	**94.12**	64.71	76.47	88.24	41.18
Pen	**88.24**	41.18	70.59	**88.24**	35.29
Cables / Towels	**94.12**	29.41	**94.12**	76.47	29.41
Bowl	**100**	64.71	88.24	**100**	82.35
Toys	**88.24**	58.82	76.47	**88.24**	64.71
Tools	82.35	47.06	**94.12**	76.47	35.29
Cone	**94.12**	64.71	64.71	70.59	41.18
Lego Structures	**70.59**	47.06	58.82	58.82	35.29
Average	**88.97**	52.21	77.94	80.88	45.59
Standard Deviation	**9.13**	13.13	13.25	12.87	18.27

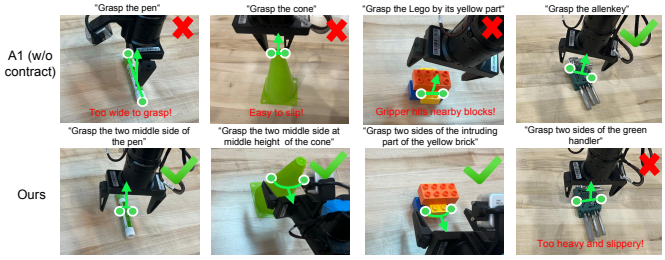

Fig. 3: Comparison of A1 (no contract) and ManiSkillFormer

gains on cones (29.41%), followed by pens (17.65%). As illustrated in Figure 3, without an explicit contract specifying *where* to grasp, perception can return distinctive edge or corner landmarks that are unsuitable for stable execution. The reason why A2 (independent generation) performs very close to the full pipeline is that for this task suite, we only test on `Pick` and `Place`, which do not have much geometry dependencies temporally.

Semantic Grounding Lego structures and toys illustrate how semantics help localize keypoints. A task instruction l such as "pick the Lego by the yellow part" or "pick the penguine by its head" names a specific semantic target. ManiSkillFormer's `Pick` contract request guides the promptable 3D perceptiom module to extract the exact semantic parts that meet both the task requirement and the skill execution requirement; the A1 ablation, lacking this explicit request, often returns less constraint keypoints that is hard for robot to execute. π_0 fails here because with only 50 demos per category the learned policy does not generalize to specific semantic request or unseen configurations.

Where contracts do not help. ManiSkillFormer underperforms the A1 ablation on *Tools* (82.35 vs. 94.12). For Tools like allenkey, the contract requests a grasp at the handle center, and because the surface of the handle is slippery, and the allenkey has a relatively heavy weight compared to the gripper itself, the robot would drop the allenkey after picking up. But for the A1 ablation, it returns two visually salient keypoints at the edge of the handle, which unexpectedlly forms a more stable grasp.

2) Q2 (Functional Manipulation)

Table III evaluates functional manipulation beyond pick-and-place, including *unscrew*, *pour*, *press*, and *fold*. These tasks examine both skill-specific geometric grounding and the coordination of atomic skills within meta skills.

Effect of skill-specific grounding. As illustrated in Fig-

TABLE III: Functional manipulation success rates (%).

Task	Ours	MOKA	A1	A2
Unscrew button	**76.47**	23.53	58.82	52.94
Pour	**82.35**	47.06	70.59	58.82
Fold cloth edge to center	**64.71**	58.82	35.29	52.94
Press button	**76.47**	**76.47**	70.59	**76.47**

ure 4, different skills applied to the same object may require different semantic keypoints. Geometric contracts explicitly specify these object-centric requirements, guiding the promptable 3D perception module to identify interaction locations that satisfy both skill-level and task-level needs. This explains why ManiSkillFormer outperforms A1 (no contract) on three of the four tasks. MOKA underperforms on unscrewing and pouring because it cannot reliably generate the skill-specific rotational trajectories required by these tasks. All methods achieve the same success rate on *Press Button*, likely because the button center is relatively straightforward to localize and pressing requires only a short execution sequence, leaving limited room for skill-specific grounding to improve performance.

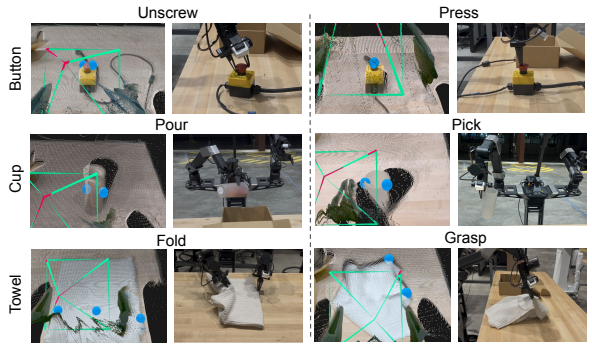

Fig. 4: Skill-dependent geometric primitives for the same object. Left Column is the 3D GLB visualization from the 3D perception module; Right Column demonstrates the real skill execution performed by AgenSkillGraph. Even for same object, different skills require different semantic keypoints.

Effect of joint generation. Compared with A2 (independent generation), ManiSkillFormer improves by 23.53% on `unscrew` and `pour`, and by 11.77% on `fold`. As illustrated in Figure 5, geometric choices for one constituent skill must also support the requirements of subsequent skills. Jointly generating skills and contracts helps coordinate these choices within a meta skill, reducing mismatches between independently specified interaction geometry and execution requirements. The gains over A2 suggest that specifying a contract for each atomic skill is not always sufficient; compatibility between constituent skills also matters.

3) Q3 (Long-horizon performance)

Except from composing simple atomic skills into complex skills, we can also compose skills (either atomic or meta) into long-horizon behabiviors. Table IV evaluates long-horizon task performances, where errors may accumulate due to dependencies between successive skills. The tasks include repeated pick-and-place across six different objects,

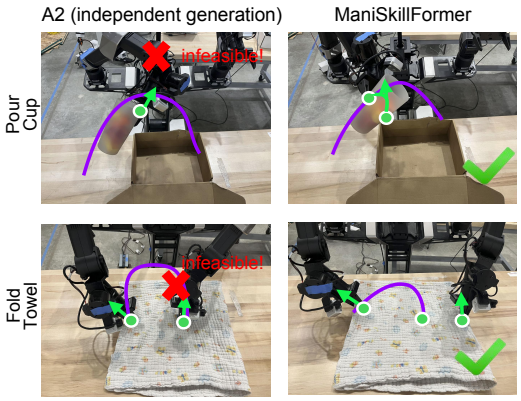

Fig. 5: Comparison of A2 pipeline and ManiSkillFormer.

cone stacking requiring precise geometric alignment, and a multi-stage manipulation sequence with strong dependencies between successive skills. The results demonstrate ManiSkillFormer's ability to compose skills into long-horizon behaviors. For simplicity, each step corresponds to an atomic skill, except for the meta skills `Pick` and `Place`, which are each counted as a single step.

TABLE IV: Long-horizon task success counts. Success requires completing the entire sequence without a failure.

Task	Ours	MOKA	A1	A2
Pick and Place 6 objects	**6/10**	2/10	4/10	**6/10**
Stack cones	**8/10**	3/10	5/10	**8/10**
Task C (11 steps)	**5/10**	1/10	2/10	1/10

Overall sequence completion. ManiSkillFormer achieves the highest or joint-highest success count on all three tasks. A2 matches ManiSkillFormer on the first task and cone stacking tasks because for each step (`Pick` or `Place`), they do not require much geometry dependencies. And ManiSkillFormer outperforms A1 (no contract) ablation on cone stacking task because this task requires precise geometry estimation, which can be improved by the explicit contract. In general, these results suggest that explicit contracts support execution across the evaluated long-horizon sequences.

Reliability under dependency. Figure 6 plots survival rate after each step in Task C (11 steps). All methods start with comparably high success rates, but the MOKA baseline and the ablated pipelines declines sharply as errors accumulate, while ManiSkillFormer maintains higher survival rates throughout, indicating that contract-based grounding produces more reliable intermediate states rather than merely more reliable individual steps.

The difference becomes evident in steps 7-9, corresponding to the meta skill *Pour*, which decomposes into the atomic skills *Grasp*, *MoveTo*, and *Rotate*. Successful "pouring" requires "grasping" the cup along two opposite sides of the cup body to provide stable support during "rotation". If the grasp is placed on edge points with a vertical direction instead, the subsequent *Rotate* skill cannot generate the correct pouring motion. Without geometric contracts, the ablated pipeline frequently selects such future-unaware grasp locations, which leads to a sharp drop in success rate during the *Rotate* stage.

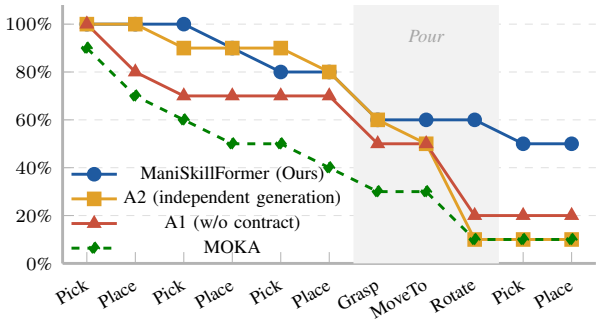

Fig. 6: Average survival rate over a 11-step long-horizon task C ("Pick yellow lego and blue lego into cup, pick red lego into red bowl, pour cup into the box, pick red bowl and place into box"). Each step corresponds to a skill. ManiSkillFormer with geometric contracts maintains higher survival rates and slows the accumulation of execution errors.

C. LLM Token Usage

Offline, the contract and skill agents consume approximately 1,608 and 2,525 tokens per skill–object request pair, respectively; their outputs are reused across executions. Online, 2D keypoint prediction uses approximately 1,518 tokens per request with a contract and 1,429 without. MOKA uses approximately 1,255 tokens per task.

VI. CONCLUSION AND FUTURE WORK

We presented ManiSkillFormer, motivated by the idea that reusable manipulation skills can be specified semantically before their geometry is grounded in a particular scene. Starting from human-defined skill schemas, LLM agents adapt motion templates to skill–object pairs and their task context, while geometric contracts make explicit what perception must provide for execution. These contracts then guide promptable 3D perception to recover the interaction geometry needed to turn the templates into robot motions. Experiments on the Galaxea R1-Lite show how this connection supports generalization across object instances, functional manipulation, and long-horizon composition.

For limitations, the current framework is limited to a predefined skill vocabulary and does not explicitly address environmental collision avoidance, dexterous manipulation, or compliance control. Future work will integrate collision-aware motion planning and richer control strategies, and investigate skill abstractions that at the same time incorporate human knowledge while adapting to broader objects and tasks and also do automatic skill discovery. The modular interface also provides a path to incorporating advances in multimodal perception (like GPT-6) while retaining reusable skill definitions.

VII. ACKNOWLEDGEMENT OF AI TOOL USAGE

GPT-5.6 and Claude assisted with text polishing of the manuscript. All AI revisions were reviewed and verified by the authors, who take full responsibility for the final content.

REFERENCES

[1] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[2] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "*pi*_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[3] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[4] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 9493–9500.

[5] I. Mishani, Y. Shaoul, and M. Likhachev, "Mosaic: A skill-centric algorithmic framework for long-horizon manipulation planning," *arXiv preprint arXiv:2504.16738*, 2025.

[6] B. Li, P. Yu, Z. Tang, H. Zhou, Y. Sun, R. Liu, and C. Liu, "Nesypack: A neuro-symbolic framework for bimanual logistics packing," *arXiv preprint arXiv:2506.06567*, 2025.

[7] C. Agia, T. Migimatsu, J. Wu, and J. Bohg, "Stap: Sequencing task-agnostic policies," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7951–7958.

[8] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini *et al.*, "Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models," in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2025, pp. 91–104.

[9] W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali, A. Mousavian, and D. Fox, "RoboPoint: A vision-language model for spatial affordance prediction in robotics," in *Conference on Robot Learning (CoRL)*, 2025, pp. 4005–4020.

[10] M. Dabhi, I. Gill, L. A. Jeni, and S. Lucey, "2d-lfm: Lifting foundation model without 3d supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026, pp. 34 303–34 311.

[11] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "VoxPoser: Composable 3d value maps for robotic manipulation with language models," in *Conference on Robot Learning (CoRL)*, 2023, pp. 540–562.

[12] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," *Proceedings of The 8th Conference on Robot Learning*, pp. 4573–4602, 2025.

[13] H. Huang, F. Lin, Y. Hu, S. Wang, and Y. Gao, "Copa: General robotic manipulation through spatial constraints of parts with foundation models," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 9488–9495.

[14] W. Tang, J.-H. Pan, Y.-H. Liu, M. Tomizuka, L. E. Li, C.-W. Fu, and M. Ding, "GeoManip: Geometric constraints as general interfaces for robot manipulation," *arXiv preprint arXiv:2501.09783*, 2025.

[15] K. Fang, F. Liu, P. Abbeel, and S. Levine, "MOKA: Open-world robotic manipulation through mark-based visual prompting," in *Robotics: Science and Systems (RSS)*, 2024.

[16] M. Pan, J. Zhang, T. Wu, Y. Zhao, W. Gao, and H. Dong, "OmniManip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 17 359–17 369.

[17] P. Sundaresan, S. Belkhale, D. Sadigh, and J. Bohg, "KITE: Keypoint-conditioned policies for semantic manipulation," in *Proceedings of The 7th Conference on Robot Learning (CoRL)*, 2023, pp. 1006–1021.

[18] X. Fang, B.-R. Huang, J. Mao, J. Shone, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, "KALM: Keypoint abstraction using large models for object-relative imitation learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[19] B. Jiang, Y. Wu, W. Zhou, C. Paxton, and D. Held, "HACMan++: Spatially-grounded motion primitives for manipulation," in *Robotics: Science and Systems (RSS)*, 2024.

[20] W. Mao, W. Zhong, Z. Jiang, D. Fang, Z. Zhang, Z. Lan, H. Li, F. Jia, T. Wang, H. Fan *et al.*, "RoboMatrix: A skill-centric hierarchical framework for scalable robot task planning and execution in open-world," 2024.

[21] OpenAI, "GPT-5.6 Sol Model," https://developers.openai.com/api/docs/models/gpt-5.6-sol, accessed: September 12, 2026.