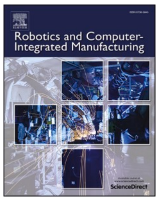
Robotics and Computer–Integrated Manufacturing 98 (2026) 103161

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim



Towards zero-shot robot tool manipulation in industrial context: A modular VLM framework enhanced by multimodal affordance representation

Qi Zhou, Yuwei Gu, Jiawen Li, Bohan Feng, Boyan Li, Youyi Bi *

Global College, Shanghai Jiao Tong University, Shanghai, China

A R T I C L E I N F O

Keywords:
Robot tool manipulation
Vision language model
Visual prompt
Affordance representation

A B S T R A C T

Robot tool manipulation in industrial context requires precise spatial localization, stable force control, and versatile adaptability across diverse tools and tasks. Traditional robot manipulation methods usually struggle to generalize to unseen scenarios and maintain reliable, precise interactions under complex physical constraints. Recent Vision Language Model (VLM)-based approaches demonstrate better generalization ability, but they often lack fine-grained modeling of spatial and force constraints that are critical for real-world industrial applications. To address these challenges, we propose a novel framework for zero-shot robot tool manipulation in industrial environments, named as *ToolManip*. This framework adopts a modular and multi-agent VLM architecture. It decomposes the manipulation process into four specialized modules—task understanding and planning, affordance reasoning, primitive reasoning, and execution monitoring—each handled by a dedicated VLM agent. To enhance the manipulation accuracy, we develop a multimodal affordance representation method that models spatial and force constraints separately. Spatial constraints are encoded via hierarchical region extraction and structured interaction fields to define keypoints and interaction directions, while force constraints are represented through force control primitive reasoning to enable precise and compliant motion and force planning. Additionally, an integrated execution-monitoring pipeline improves the system reliability by tracking the status of each task step and performing stepwise corrections. Experimental results demonstrate that ToolManip achieves robust, generalizable, and high-accuracy performance in various constraint-rich industrial tool manipulation tasks. Our work contributes to the development of advanced robotic manipulation methods for industry and smart manufacturing environments empowered by generative artificial intelligence.

1. Introduction

Recent advancements of embodied intelligence systems such as Tesla's *Optimus* and Boston Dynamics' *Atlas* have demonstrated their ability to perform basic manipulation tasks, such as bottle cap screwing and box moving [1]. These developments mark a significant step toward more autonomous and versatile robotic capabilities beyond merely imitating human-body motions. However, these robotic platforms are still limited in performing flexible tool manipulations in real-world industrial and manufacturing settings [2]. Compared with typical household tasks, industrial tool manipulation presents significantly higher requirements. On one hand, the wide variety of task and tool types calls for better generalization ability, and robots must be able to autonomously identify suitable tools, plan appropriate manipulation actions, and execute them with high accuracy [3]. On the other hand, industrial tasks often involve strict spatial constraints, complex physical interactions and precise force control [4]. For example, operations such as scraping putty with a blade or cleaning surfaces with a roller (see Fig. 1) require not only accurate spatial alignment between tools and target objects, but also stable and compliant force execution to ensure safe operations.

Today in factories, many of these operations are still predominantly performed manually or rely on pre-programmed robots. The scalability and autonomy of robot tool manipulation in complex industrial settings are limited. To address this challenge, researchers have developed a number of robot tool manipulation methods, mainly including Imitation Learning, optimization, and Reinforcement Learning based methods [5–7]. Imitation Learning allows efficient policy acquisition from expert demonstrations, but struggles to generalize across unseen tasks and tools [8]. Optimization-based methods can yield high-precision control given

* Corresponding author.
E-mail addresses: zhouqi1998@sjtu.edu.cn (Q. Zhou), yuwei.gu@sjtu.edu.cn (Y. Gu), david_lee@sjtu.edu.cn (J. Li), bohan.feng@sjtu.edu.cn (B. Feng), liboyan@sjtu.edu.cn (B. Li), youyi.bi@sjtu.edu.cn (Y. Bi).

https://doi.org/10.1016/j.rcim.2025.103161
Received 29 June 2025; Received in revised form 23 September 2025; Accepted 30 September 2025
Available online 7 October 2025

accurate geometric and dynamic models, yet they are sensitive to model inaccuracies and environmental variations [9]. Reinforcement Learning excels in learning control strategies in dynamic or unstructured environments but suffers from high training cost and sim-to-real gaps [10].

Recently, achievements in generative artificial intelligence such as large language models (LLMs) and vision language models (VLMs) have emerged as promising techniques for robot task planning and manipulation strategy generation, offering strong reasoning and generalization capabilities [11]. Trained on large-scale datasets, VLMs encode rich general and domain knowledge, enabling them to infer tool-use intention, identify target objects, and propose feasible robot actions from visual and textual inputs. Although these models demonstrate promising task inference capabilities, their applications remain primarily confined to household scenarios with limited precision requirements and weak physical constraints. To support reliable robot tool manipulation in industrial environments, a more systematic solution integrating task understanding, tool recognition, and precise execution is needed.

To enable precise robot tool manipulation, affordance modeling serves as a fundamental role. It aims to represent the potential interactions between the robot and its environment, typically characterized by spatial locations, interaction directions or force constraints [12]. Existing affordance modeling approaches can be broadly divided into two categories. The first relies on supervised learning to predict relevant interaction regions from visual input, but often demands extensive labeled data and struggles to generalize across unseen tools or tasks [13]. The second category leverages VLMs to infer interaction regions by reasoning from task descriptions and visual scenes, enabling better generalization to novel tasks [14]. However, these methods primarily capture coarse positional cues, such as keypoints or bounding boxes, while lacking explicit modeling of directional interactions and force constraints. This limits their applicability in industrial scenarios that demand high precision, stability, and fine-grained physical control such as the putty removal or nail hammering tasks (see Fig. 1).

To address these challenges, we propose a novel robot tool manipulation framework for industrial environments that integrates multiple VLM agents named as ToolManip. It decomposes the tool manipulation process into four functionally distinct modules—task understanding and planning, affordance reasoning, primitive reasoning, and multi-stage execution, each assigned to a specialized VLM agent—Task Manager, Affordance Analyst, Motion Planner, and Execution Inspector. Each agent in ToolManip is equipped with agent-specific memory, enabling specialized reasoning while ensuring consistency and stability across multi-stage tasks. These agents collaboratively interpret multimodal inputs, infer task constraints and generate executable robot actions with step-level action monitoring. Particularly, we develop a multimodal affordance representation method to support these agents to localize task-relevant regions and infer interaction constraints through structured spatial representations. We construct a candidate set of interaction points and directions through structured visual prompts, and leverage a motion/force primitive library to assist the VLM agent with reasoning about spatial and physical constraints. This joint modeling of geometry and contact constraints enables precise manipulation and stable force control in complex tool-use tasks. Moreover, by incorporating multimodal reasoning with online step verification and adaptive correction, ToolManip achieves robust, generalizable, and zero-shot tool manipulation performance.

The main contributions of this work include:

- A zero-shot robot tool manipulation framework for industrial environments based on modular VLM architecture is proposed. It assigns the work of task understanding, affordance and primitive reasoning, as well as task monitoring into dedicated VLM agents equipped with agent-specific memory. This framework enables modular, context-aware tool manipulation with enhanced robustness and generalizability across industrial scenarios.
- A multimodal affordance representation method that incorporates both spatial and physical constraints is developed. It models keypoints and interaction directions via hierarchical region extraction and structured interaction fields, and encodes force constraints through force primitives to enable precise motion and force planning.

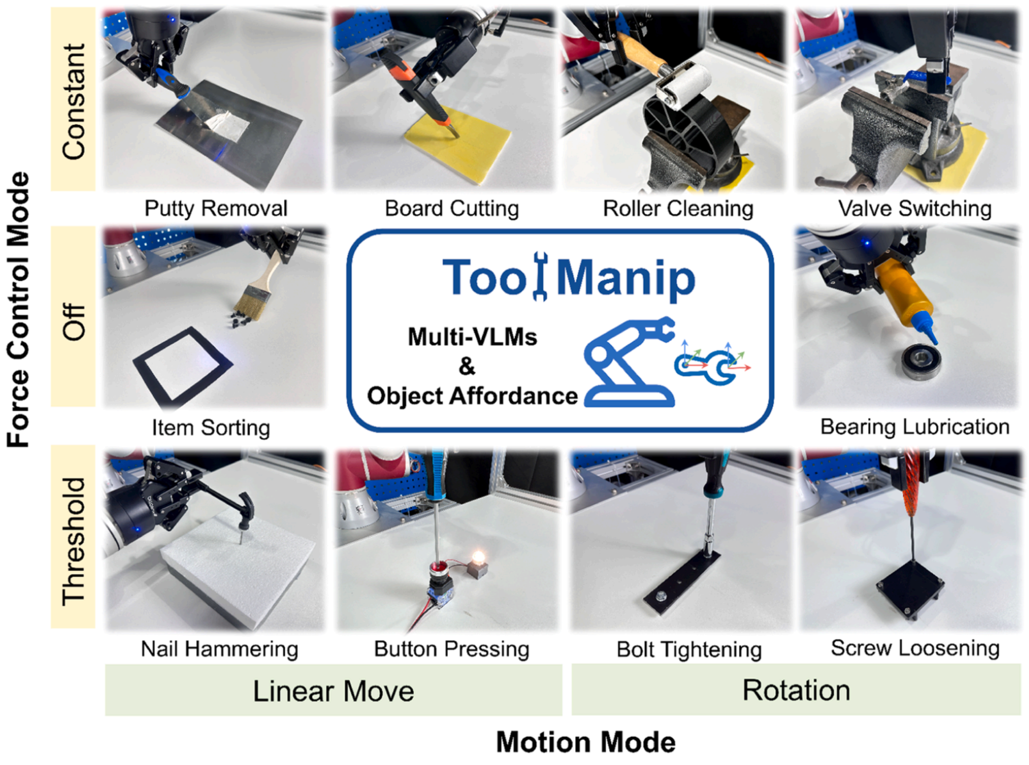


Fig. 1. Typical open-vocabulary tool manipulation tasks in industrial context.

- An execution-monitoring pipeline for tool manipulation is designed to mitigate the VLM hallucinations. It incorporates online status tracking and stepwise correction, enabling stable and adaptive manipulation in constraint-rich tasks.

The rest of this paper is organized as follows. Section 2 reviews related work in robot tool manipulation strategies, the application of foundation models in robotics and affordance modeling techniques. Section 3 introduces the architecture of the proposed framework and explains its key components. Section 4 presents the comparison and ablation experimental setups to validate our approach, and provides a detailed discussion on the experimental results. Section 5 summarizes our work and highlights potential future research directions.

2. Related work

2.1. Robot policy for tool manipulation

Tool manipulation is a fundamental capability for robots to perform complex tasks as well as a key indicator of robot intelligence [15]. Existing research on robot control policies for tool manipulation can be broadly categorized into four representative paradigms: imitation learning, optimization-based methods, reinforcement learning, and reasoning leveraging large-scale pre-trained models.

Imitation learning enables robots to learn from human or expert demonstrations, offering an efficient pathway to acquire high-quality manipulation skills [3,5,8,16,17]. For example, Seita et al. [5] proposed ToolFlowNet, a neural network that learns continuous tool manipulation by predicting per-point flow to derive robot actions, achieving effective imitation learning for tasks like scooping and pouring. Tang et al. [17] presented a tool manipulation approach that infers tool affordances from a single demonstration to enable one-shot task execution. However, the effectiveness of imitation learning depends on the diversity and coverage of high-quality demonstration data, which limits its generalizability to unseen tasks and environments.

Optimization-based methods explicitly formulate geometric and dynamic models of tools and solve for manipulation trajectories through numerical optimization [6,18–22]. For instance, Zhang et al. [20] proposed a framework that identifies essential physical properties of tool-use via Finite Element Method (FEM) and symbolic regression, and formulates an optimal control strategy using a Virtual Kinematic Chain model to generate effective manipulation plans. Although these methods can achieve high precision when accurate models are available, they are often sensitive to dynamic environmental changes, variations in tool geometry and modeling accuracies.

Reinforcement learning (RL) allows robots to autonomously acquire tool-use skills through trial-and-error interactions with the environment, which is particularly advantageous in dynamic and hard-to-model scenarios [7,23,24]. For example, Liu et al. [7] developed a deep RL framework that jointly optimizes tool morphology and robot manipulation strategies, enabling adaptive tool use across diverse tasks. Hiranaka et al. [23] presented a hybrid approach that integrates human feedback with primitive skill-based RL to safely and efficiently train robots for complex long-horizon manipulation tasks. Nonetheless, deploying RL models still faces challenges such as long training time and difficulties in transferring policies from simulation to the real world.

Recently, researchers have explored using large pre-trained models, such as large language models (LLMs) and vision-language models (VLMs), to reason tool-use strategies directly from high-level language instructions or visual inputs, thereby reducing the reliance on extensive annotated demonstrations [15,25]. Compared to traditional imitation learning, optimization or reinforcement learning based methods, large model-based methods offer stronger generalization across diverse tasks, which represent a promising paradigm for scalable and flexible robotic manipulation in open-world settings. For instance, Car et al. [25] combined LLM with affordance reasoning to generate language-guided manipulation plans, which supports robots in performing complex tool-use tasks in dynamic environments. Xu et al. [15] demonstrated the capability of LLM to infer novel tool-use strategies and generate executable code for creative and unstructured robotic tasks. However, these approaches have mainly been tested in household settings and remain insufficient for the tool manipulation in industrial context with high accuracy and robustness requirements. In this work, we expect to fill this gap by developing a tool manipulation framework that combines large-model reasoning with low-level motion and force planning, enabling scalable, robust, and high-precision operations in diverse industrial tasks.

2.2. Foundation models in robotics

Recent advances in foundation models, such as LLMs and VLMs, have significantly enhanced robots' capabilities in natural language understanding, multimodal perception, and cross-task generalization [11]. Some studies explore to fine-tune foundation models on domain-specific datasets and endow them with specialized reasoning and perception capabilities, such as 3D spatial understanding [26] and physical property inference [27]. However, fine-tuning is often constrained by the high costs of data collection and model training.

Alternatively, researchers also attempt to directly leverage foundation models to support different stages in robotic task execution, such as high-level task planning [9,28–30] and action generation [31,32]. Recent pipelines further incorporate the reasoning capabilities of foundation models in multiple stages of the tool manipulation workflow. For example, Lai et al. [33] employed an LLM to process multimodal voice and human posture inputs, resolve contextual ambiguities, generate structured robot action sequences and enforce output constraints to ensure safe and robust human–robot interaction. Similarly, Fan et al. [10] proposed an LLM-driven framework that utilizes LLM to extract process parameters, design and verify tool paths, and perform high-level task reasoning to enable adaptive industrial robot control. However, these methods often utilize a single central large model to handle multiple tasks, and such tightly coupled implementations often lack modularity and offer limited flexibility for adapting to diverse and dynamic task requirements.

To overcome these limitations, recent research has explored multi-agent VLM frameworks in which multiple specialized agents collaboratively handle distinct sub-tasks [34–37]. For example, Mei et al. [34] proposed a framework that leverages three VLM agents for visual perception and task planning, while integrating both internal and external error correction mechanisms to detect and replan actions when execution failures occur, thereby enhancing the reliability of robotic task execution across multiple stages. Wang et al. [35] proposed a multi-agent Vision Large Language Model (VLLM) framework for robotic planning, in which four specialized agents collaboratively handle various aspects of demanding robotic tasks from high-level robotic planning to low-level position-based control. Despite these advances, most existing frameworks still fall short in robust spatial reasoning and fine-grained constraint modeling, both of which are essential for high-precision industrial tool manipulation. For instance, robust spatial reasoning allows the system to accurately identify and align interaction regions between tools and operational objects. Fine-grained constraint modeling captures task-specific positional and physical interaction requirements, which are critical for stable and safe execution in constrained environments. In our work, we expect to propose a modular multi-agent VLM framework enhanced by spatial reasoning and constraint representation, offering a more interpretable and robust solution for tool manipulation in high-precision industrial settings.

2.3. Affordance modeling for robot control

Affordance in robotics characterizes the possible interactions between robot and environment, such as graspability, pushability, and tool usability. Recent studies have demonstrated that representing affordances using keypoints and bounding boxes provides a compact and structured encoding of task-relevant interaction regions, which facilitates task reasoning, motion planning, and manipulation control [9,10,38]. Existing affordance modeling methods can be categorized into learning-based prediction and pre-trained large model-based reasoning approaches. Learning-based methods rely on training customized neural networks to predict affordance keypoints or regions. For example, kPAM [12] and KETO [13] employ deep neural networks to learn keypoint distributions for various objects. Ju et al. [39] developed a general framework, Robo-ABC, which leverages affordance memories extracted from large-scale human videos and transfers contact points to novel objects for zero-shot manipulation. However, these methods typically require extensive annotated datasets and face challenges in generalization and reusability.

In contrast, pre-trained large model-based reasoning methods exploit the visual-language reasoning capabilities of VLMs to perform structured affordance modeling and planning without additional supervision [14,32,40–43]. For instance, Nasiriany et al. [14] presented a novel visual prompting approach for VLMs, which iteratively generates and refines candidate interaction points or arrows through multi-round visual-language reasoning. This approach enables VLMs to perform zero-shot spatial inference and low-level robotic control by selecting and optimizing visual proposals (e.g., actions or trajectories) without requiring task-specific fine-tuning. Lei et al. [43] proposed Scaffolding method that overlays structured dot grids and explicit coordinates as visual anchors to guide VLM-based spatial reasoning. Researchers also explore to combine visual object detectors (e.g., SAM [44], GroundingDINO [45]) with VLMs to extract object-specific regions and candidate interaction features. Representative methods include MOKA [41], which integrates GroundedSAM [46] for segmentation and applies farthest point sampling on mask contours to generate candidate keypoints, while the area within the mask's interior is neglected. In addition, OmniManip [42] extends affordance representations by introducing an extra interaction direction to bridge VLM's high-level reasoning with precise robotic manipulation. However, these directions are limited to principal axes within canonical coordinate system of objects, restricting precision for complex manipulations.

In summary, current pre-trained large model-based methods mainly cover positional constraints but overlook accurate interaction direction modeling and stable force control, which are crucial for reliable industrial tool manipulation scenarios. For instance, direction constraints are essential in tasks such as cutting boards or tightening bolts (see Fig. 1), where the tool must operate along a specific angle against the operational object. Similarly, stable force control is also critical in tasks like putty removal and roller cleaning, which require continuous and consistent contact force to ensure successful task execution. To address these limitations, we expect to propose a multimodal affordance modeling approach that incorporates both precise positional and force constraints, where keypoints and interaction directions serve as fine-grained representations of the spatial interactions, providing a generic constraint formulation for accurate and robust tool manipulation.

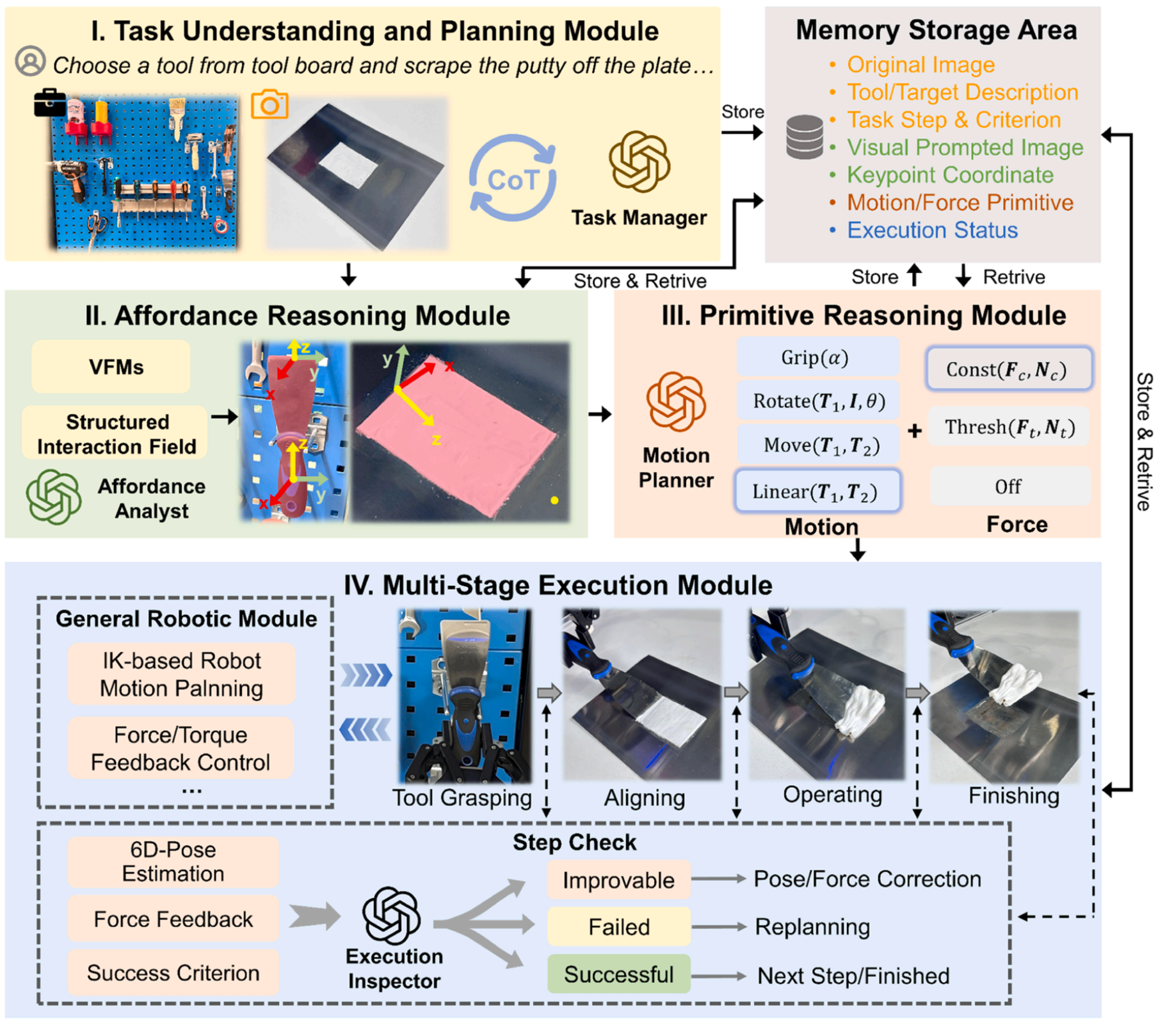


Fig. 2. Overall structure of proposed robot tool manipulation framework (ToolManip).

3. Methods

3.1. Overall structure of the proposed approach

Fig. 2 presents the overall structure of the proposed robot Tool Manipulation Framework (ToolManip),[1] which consists of four main modules: Task Understanding and Planning, Affordance Reasoning, Primitive Reasoning and Multi-Stage Execution. Each module integrates a specialized VLM agent (i.e., Task Manager, Affordance Analyst, Motion Planner and Execution Inspector) to handle planning, reasoning or examining jobs at different stages of successfully completing a tool manipulation task. In addition, a unified Memory Storage Area ensures seamless information storing and retrieving across the entire pipeline.

Specifically, in the Task Understanding and Planning module, the Task Manager agent interprets users' manipulation instructions (e.g., *scrape the putty off the plate*) and visual scene inputs to identify the suitable tools, detect task-relevant objects and generate step-by-step task plans following a chain-of-thought (CoT) reasoning paradigm. Then, the Affordance Reasoning module constructs structured visual prompts, which enable the Affordance Analyst agent to infer fine-grained positional constraints, including key interaction points and directions required for the manipulation. After that, the Primitive Reasoning module employs the Motion Planner agent to map task descriptions and inferred affordance constraints to appropriate motion and force primitives retrieved from a pre-defined primitive library. These primitives are used to formulate low-level robot control commands for each step of the manipulation sequence. Finally, in the Multi-Stage Execution module, the robot executes the planned primitives step by step. After each execution step, the Execution Inspector agent assesses the actual outcomes using multimodal feedback—including both positional deviation and force sensor readings—and applies adaptive corrections, such as minor trajectory adjustments or full replanning when failures are detected. This execution-monitoring mechanism is essential for ensuring robust task performance and mitigating potential hallucinations from VLM outputs. Throughout the entire process, the Memory Storage Area manages intermediate data exchange among all modules to ensure data consistency and seamless inter-module interaction (e.g., original images and tool/target description for affordance reasoning, keypoint coordinates and task steps for primitive reasoning and planned primitives for robot execution).

The proposed framework employs a modular VLM architecture that decouples task understanding, affordance reasoning, motion planning and monitoring into context-dependent stages. By explicitly allocating each VLM agent to a specialized sub-task, the system mitigates reasoning ambiguity often observed in single-VLM agent paradigms, resulting in more robust and precise tool-use performance across diverse scenarios. The prompt examples for all VLM agents are detailed in Appendix Tables A1-A4. The key techniques involved in this framework are presented in the following sections.

3.2. Task understanding and planning

In the Task Understanding and Planning module, a structured multi-stage Chain-of-Thought (CoT) reasoning strategy is employed to extract essential semantic elements for downstream execution. This process emulates human-like sequential decision-making by incrementally refining task-relevant information from visual and textual inputs. As illustrated in Fig. 2, the VLM agent Task Manager interprets user instructions alongside the workspace and tool board images to infer key task attributes and generate an executable plan. Fig. 3 shows a representative CoT reasoning example for a putty scraping task. The agent successively identifies the target object, detects available tools, selects the most suitable tool, and formulates a multi-step task plan. This hierarchical reasoning provides crucial semantic cues for subsequent modules.

The inferred properties of the target object and the selected tool are organized as structured semantic representations:

$$N_o = \{n_o, p_o, c_o\} \tag{1}$$

$$N_t = \{n_t, p_t, c_t\} \tag{2}$$

where, N_o and N_t are the properties of object and tool, $n,\ p,\ c$ represent the name, position and color, respectively. This information serves as descriptive prompts for accurate object detection in the subsequent Affordance Reasoning module. Finally, a structured sequence of task steps S for robot execution is generated and expressed as:

$$S = \{s_i\}_{i=1}^{N} \tag{3}$$

where each step $s_i = \{e_i, a_i, o_i, \phi_i\}$ specifies the motion-executing entity (e.g., robot arm or tool) e_i, the action to be performed a_i, the target object o_i, and the success criterion ϕ_i for each step. The original scene and tool images, along with all intermediate VLM reasoning results (e.g., descriptions of tool and object, task steps and success criteria) are stored in the Memory Storage Area.

3.3. Visual prompt enhanced affordance reasoning

In ToolManip, accurately localizing the relevant operational regions of both the target object and the tool is critical for successful task execution. To this end, we introduce keypoints and interaction vectors as the positional affordance representation form and design a structured visual prompt generation and affordance reasoning method, as illustrated in Fig. 4. The keypoints encode the spatial positions of interaction regions, while the interaction vectors specify their directional constraints, together defining the robot's essential pose states during task execution. The overall workflow proceeds as follows: the regions of interest (ROIs) for both the tool and target objects are first extracted through a Hierarchical Region Extraction (HRE) process. Next, a Structured Interaction Field (SIF) is constructed and annotated on the images, serving as visual prompt cues for Affordance Analyst (the VLM agent) to reason about the affordance constraints more effectively. Finally, the keypoints and directions are computed using the SAM-6D [47] model. The implementation details of our method are provided in the following subsections.

3.3.1. Structured visual prompt generation

As shown in Fig. 4, the first step in structured visual prompt generation is to accurately identify the ROIs for both the tool and the target object within the input images. To achieve fine-grained and task-relevant localization, we develop a Hierarchical Region Extraction (HRE) mechanism that progressively localizes ROIs from coarse to fine scales. First, the original images I and corresponding tool and task descriptions N_1, provided by the Task Manager, are input to the LangSAM [48] model. LangSAM combines the zero-shot segmentation capability of SAM with the text-grounding power of GroundingDINO to perform text-to-bounding-box object detection and segmentation without additional training. The ROIs extraction process can be formulated as follows:

$$\{s_1, m_1\} = LangSAM(I, N_1) \tag{4}$$

where s_1, m_1 denote the bounding box coordinates and segmentation masks of the target object and tool.

The bounding box area is slightly enlarged by an expansion factor a to contain background information:

$$s_1' = s_1(1 + a) \tag{5}$$

[1] For the source code of this project, please visit: https://github.com/toolmanip/ToolManip.

CoT Step1: Task Understanding *Identify the target object of putty scraping task based on the task image. Format: {Object name, Position, Color}*	*Putty, Center of image, White*
CoT Step2: Tool Identification *List all visible tools on the tool board based on the tool image.*	*Brush, Scraper, Roller, Hammer, Knife, Paint, Wrench, Electrical Tape,…*
CoT Step3: Tool Selection *Select the best tool for putty scraping. Format: {Tool name, Position, Color}*	*Scraper, Right side of the tool board, Silver blade with black and blue handle*
CoT Step4: Task Planning *Describe task steps and success criterions for a robot with a two finger gripper. Format: {Step: subject, action, object; success criterion}*	*Step1: Gripper holds scraper handle; scraper handle is firmly grasped. Step2: … Step3: …*

Fig. 3. An example of CoT prompt design. The left side lists the prompt instructions for each step (Task Understanding, Tool Identification, Tool Selection and Task Planning), and the right side shows the corresponding outputs generated by the VLM agent, Task Manager.

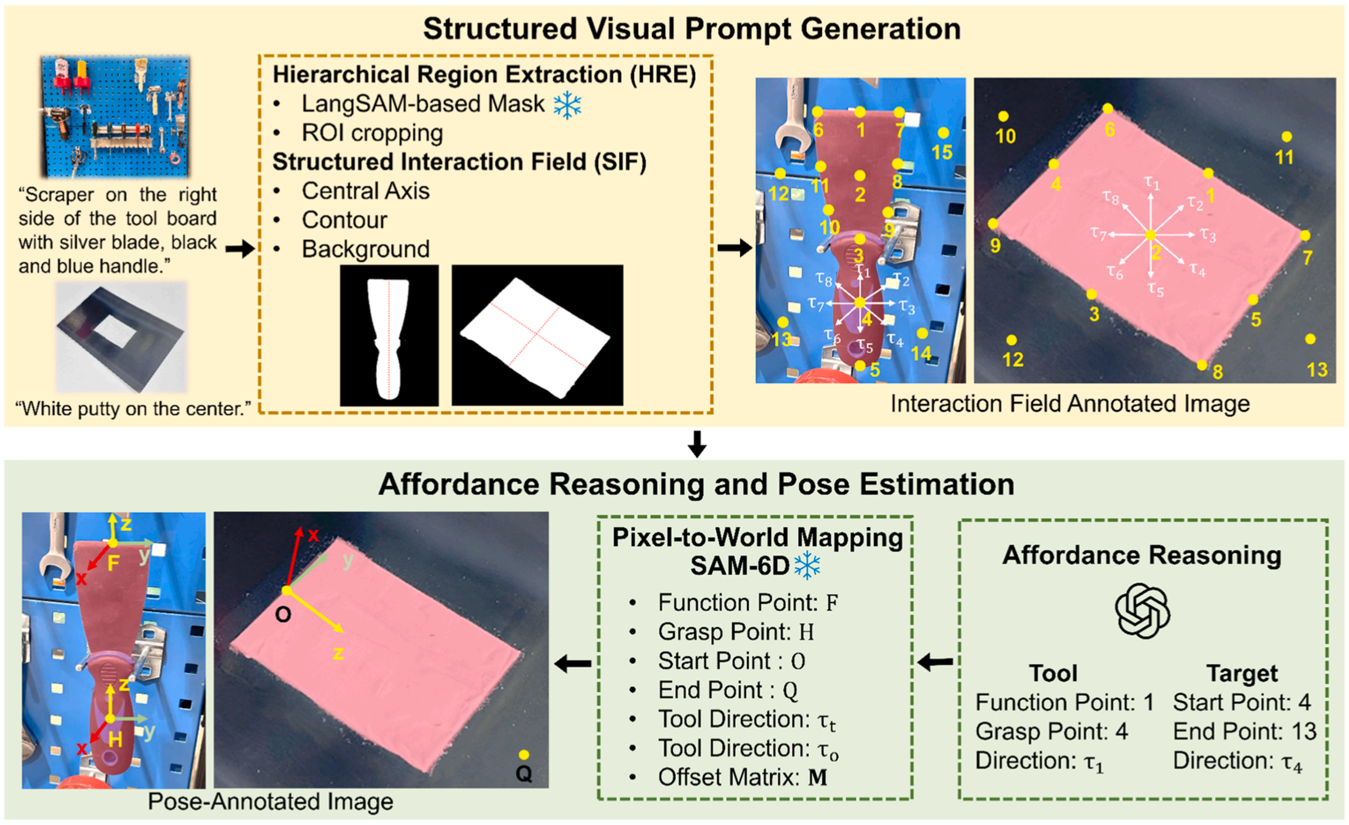

Structured Visual Prompt Generation
"Scraper on the right side of the tool board with silver blade, black and blue handle."
"White putty on the center."
Hierarchical Region Extraction (HRE)
• LangSAM-based Mask
• ROI cropping
Structured Interaction Field (SIF)
• Central Axis
• Contour
• Background
Interaction Field Annotated Image
Affordance Reasoning and Pose Estimation
Pose-Annotated Image
Pixel-to-World Mapping SAM-6D
• Function Point: F
• Grasp Point: H
• Start Point : O
• End Point : Q
• Tool Direction: τ_t
• Tool Direction: τ_o
• Offset Matrix: M
Affordance Reasoning
Tool
Function Point: 1
Grasp Point: 4
Direction: τ_1
Target
Start Point: 4
End Point: 13
Direction: τ_4

Fig. 4. The architecture of visual prompt generation and affordance reasoning.

Next, a second-stage extraction is performed specifically for the tool to further isolate its functional parts (e.g., brush head) and grasping regions (e.g., brush handle):

$$\{s_2, m_2\} = LangSAM\left(I_{s_1'}, N_2\right) \tag{6}$$

where $I_{s_1'}$ denotes the cropped and slightly expanded image from the first-stage extraction, and s_2, m_2 represent the refined ROIs coordinates and masks for the tool's functional region. The segmented ROIs for both the tool and target object are then rescaled using bilinear interpolation to increase their spatial size and facilitate accurate affordance annotation in subsequent processing.

After obtaining the ROIs, we construct candidate sets of keypoints and interaction directions to serve as the affordance representation. Four keypoints and two directions are defined as spatial constraints: the grasping and functional points on the tool, the start and end points on the target object, along with directions at the tool's functional point and the target's start point that need to be aligned during tool operation.

Considering tool grasping points usually align symmetrically along with the geometric central axis, and functional points are predominantly distributed along the contour of the functional region of the tool. A Structured Interaction Field (SIF) mechanism is designed to explicitly define candidate keypoints and interaction directions based on the following guidelines:

- Extract the morphological skeletons of the masks for both the tool and the target object, and scatter candidate points symmetrically and uniformly along their central axes.
- Evenly scatter points along the contours of the tool's functional region and the target object's mask, ensuring that geometric corners are explicitly included as candidate keypoints.
- Randomly scatter extra points in the background regions to improve the robustness of the candidate set.

All keypoints are numbered sequentially from 1 to n. A standardized directional candidate set is also defined, which comprises vectors uniformly distributed in angle (denoted as directions τ_i). This ensures that the final visual prompt encodes both precise spatial location and directional information, supporting fine-grained reasoning in the subsequent affordance analysis.

3.3.2. Affordance reasoning and 6D pose estimation

Subsequently, the Keypoint Analyst infers the appropriate keypoints and direction vectors from the candidate set. Given the annotated tool image I_{pt}, target area image I_{po}, and the task steps S derived from task planner, the inferred interaction set is denoted as:

$$\{\mathrm{F}, \mathrm{H}, \mathrm{O}, \mathrm{Q}, \tau_t, \tau_o\} = keypoint(I_{pt}, I_{po}, S) \tag{7}$$

where F, H, O, Q denote the functional and grasping points of the tool, and the start and end points of the target object, respectively; τ_t, τ_o are the principal directions at the tool's functional point and the target's start point. Since only the position and a single interaction direction is insufficient to fully define the robot's 6D pose, a local coordinate frame is constructed for each keypoint. Specifically, its Z-axis is aligned with the interaction direction vector. The Y-axis is defined along the contour edge direction perpendicular to the Z-axis where both possible edge directions are considered as candidates. The X-axis is determined by the right-hand rule based on the defined Y and Z axes.

Next, the SAM-6D model is employed to estimate the 6D poses of the geometric centers of both the tool and the target object. Based on these center poses, the relative pose offset matrices **M** are computed for all keypoints, which indicate the transformation of each keypoint with respect to the corresponding geometric center of the tool or target object. During task execution, the SAM-6D model periodically estimates the 6D pose of the tool or target object's geometric center. This estimated pose, together with each keypoint's precomputed offset matrix **M**, is used to compute the current pose $\boldsymbol{T}$ of each keypoint, enabling precise and robust pose tracking throughout the manipulation process.

3.4. Motion and force primitives reasoning

After obtaining the keypoints and interaction directions, the next step is to generate generic, executable, and highly reusable robot control commands. To support this step, we define a set of standardized Motion Primitives and Force Control Primitives, as summarized in Table 1. These primitives decompose complex manipulation tasks into structured low-level instructions, reducing control complexity and improving the system's generalization capability.

The motion primitives specify the robot's basic motion patterns during task execution, including free-space movements without contact constraints, precise linear trajectories for position-constrained actions, rotations about fixed points for orientation adjustments, and gripping operations. Each primitive is defined as a parameterized function, with inputs such as start and end poses, rotation centers and angles, and gripper states, which can be flexibly adapted to diverse task scenarios and physical constraints. For example, $\mathrm{Move}(\boldsymbol{T}_1, \boldsymbol{T}_2)$ denotes unconstrained robot motion in free space from start pose $\boldsymbol{T}_1$ to end pose $\boldsymbol{T}_2$. $\mathrm{Rotate}(\boldsymbol{T}_1, \boldsymbol{I}, \theta)$ defines rotation about a defined center $\boldsymbol{I}$, starting from pose $\boldsymbol{T}_1$ with a rotation angle θ.

Table 1
Motion and force primitives.

Category	Primitive	Execution Format	Control Parameters
Motion	Free Move	$\mathrm{Move}(\boldsymbol{T}_1, \boldsymbol{T}_2)$	$\boldsymbol{T}_1$: Start Pose
	Linear Move	$\mathrm{Linear}(\boldsymbol{T}_1, \boldsymbol{T}_2)$	$\boldsymbol{T}_2$: End Pose
	Rotate Move	$\mathrm{Rotate}(\boldsymbol{T}_1, \boldsymbol{I}, \theta)$	$\boldsymbol{T}_1$: Start Pose $\boldsymbol{I}$: Rotation Center θ: Rotation Angle, $\theta \in (-360^\circ, 360^\circ)$
	Grip Move	$\mathrm{Grip}(\alpha)$	α: Grip Sign (0 = open, 1 = close)
Force	Constant Force	$\mathrm{Const}(\boldsymbol{F}_c, \boldsymbol{N}_c)$	$\boldsymbol{F}_c$: Constant Force (N) $\boldsymbol{N}_c$: Constant Torque (N·m)
	Threshold Force	$\mathrm{Thresh}(\boldsymbol{F}_t, \boldsymbol{N}_t)$	$\boldsymbol{F}_t$: Force Threshold (N) $\boldsymbol{N}_t$: Torque Threshold (N·m)
	Off	Off	None

To ensure stable contact interactions and compliant force regulation during manipulation, we also introduce a set of force control primitives, which include: (1) a constant force mode (i.e., $\mathrm{Const}(\boldsymbol{F}_c, \mathrm{N}_c)$) for maintaining steady contact force between the tool and the target surface, and (2) a threshold force mode (i.e., $\mathrm{Thresh}(\boldsymbol{F}_t, \boldsymbol{N}_t)$) that triggers specific actions when the applied force exceeds a predefined limit, enabling responsive and safe behavior during contact-rich tasks. An additional mode allows force control to be switched off when not required. The relevant force parameters are adaptively selected by the Motion Planner VLM from a predefined parameter library, ensuring consistent force stability and operational safety.

Based on the task step instructions, prompted images, and affordance information, the Motion Planner VLM agent then selects suitable combinations of motion and force primitives from the primitive library to generate a primitive sequence for each task step. These structured low-level control commands can be directly executed by the Multi-Stage Execution module, which is detailed in Section 3.5.

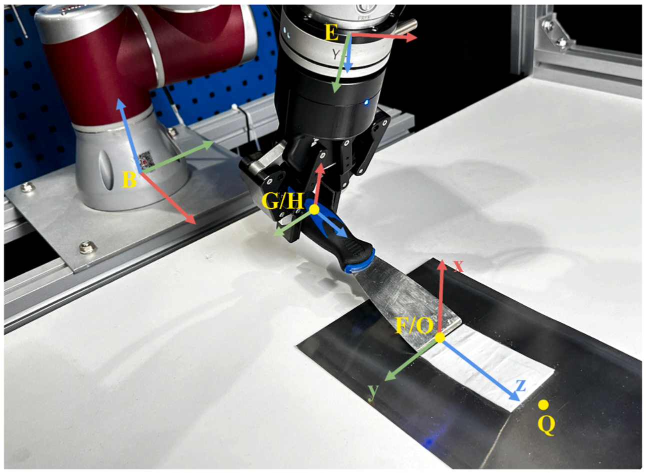


Fig. 5. Illustration of the coordinate frames used for tool grasping and tool-to-target alignment. The robotic gripper is grasping the tool and aligning its functional point with the start point on the target object.

3.5. *Online motion planning and step checking*

3.5.1. *Robot motion planning*

Robotic tool-manipulation tasks typically consist of three phases: tool grasping, tool–target alignment, and operation. As illustrated in Fig. 5, we establish a rigid-body transformation model that converts the spatial information of keypoints and interaction directions into the desired robot end-effector poses for motion planning. During the tool grasping phase, the gripper needs to align the tool's grasping point with a desired pose. This relationship is formulated as:

$$\mathbf{T}_B^H = \mathbf{T}_B^E \cdot \mathbf{T}_E^G \cdot \mathbf{T}_G^H \tag{8}$$

here the homogeneous transformation matrix $\mathbf{T}_B^E$ denotes the pose of coordinate frame E with relative to frame B. The objective in the tool grasping phase is to compute the end-effector pose $\mathbf{T}_B^E = \mathbf{T}_B^H \cdot (\mathbf{T}_E^G \cdot \mathbf{T}_G^H)^{-1}$, ensuring that the tool's grasping frame aligns with the expected pose $\mathbf{T}_B^H$.

After grasping, the robot must manipulate the tool so that its functional point frame aligns with the start point of the target object. This relationship is expressed by the transformation:

$$\mathbf{T}_B^O = \mathbf{T}_B^E \cdot \mathbf{T}_E^G \cdot \mathbf{T}_G^H \cdot \mathbf{T}_H^F \cdot \mathbf{T}_F^O \tag{9}$$

The desired end effector pose is denoted as $\mathbf{T}_B^E = \mathbf{T}_B^O \cdot (\mathbf{T}_E^G \cdot \mathbf{T}_G^H \cdot \mathbf{T}_H^F \cdot \mathbf{T}_F^O)^{-1}$, here B is robot base frame (world frame), E is robot end-effector frame, G is gripper center frame, H is tool grasping point frame, F is the tool's functional point frame and O is target object frame.

Based on these transformations, the desired end-effector pose for each task step is determined, and the corresponding robot joint configurations can be calculated via inverse kinematics (IK). To ensure smooth and continuous motion, a linear interpolation strategy is applied to generate continuous motion trajectories, which are then executed online by the robot controller.

3.5.2. *Step check*

In real-world robotic executions, environmental disturbances, grasping inaccuracies, or unstable force control can introduce motion deviations and affect task performance. To enhance system robustness and enable self-recovery, a step-checking mechanism is developed, as illustrated in Algorithm 1. Specifically, after each planned primitive execution (lines 2–4 in Algorithm 1), the system captures a new scene image I_i. The geometric center poses of the tool and target are estimated using the SAM-6D model. The poses of all keypoints are then updated based on the precomputed offset matrices and will be annotated in a step-specific prompted image I_{ri}. The multimodal deviation for the current step is defined as $E_i = \{\Delta T_i, \Delta F_i\}$, where the pose error ΔT_i is obtained by comparing each keypoint pose with its ideal pose, and the force error ΔF_i is measured by the force/torque sensor (line 5). The VLM agent Execution Inspector then evaluates the current step's execution status based on the annotated image I_{ri} and the success criterion of each task step (line 6). The evaluation results are categorized into three cases:

- **Failed** (lines 7–11): If an unrecoverable error is detected (e.g., tool dropping), the VLM agent replans the primitive sequence and parameters for both the current and preceding steps and re-executes the affected task steps. If the number of consecutive failures exceeds a predefined threshold, the task is terminated and reported as failed.
- **Improvable** (lines 12–13): If the pose and force deviation is within a correctable range (e.g., minor positional drift or torque fluctuation), the system incrementally adjusts the primitive parameters, such as end-effector pose corrections or force-control gains according to the computed error E_i. This supports online compensatory refinement without restarting the entire plan.
- **Successful** (lines 14–15): If the step state satisfies the success criterion, the robot proceeds to the next step, iterating until all the task steps are successfully executed.

This step-checking mechanism provides closed-loop monitoring and adaptive correction, significantly enhancing the online robustness and operational safety of zero-shot industrial manipulation tasks.

4. Experiments and results

4.1. *Experiment settings*

To validate the proposed approach, we conducted a series of experiments with a total of 10 typical industrial tool manipulation tasks involved as shown in Fig. 1. These tasks include both linear and rotational motion modes, with five tasks designed for each type. Based on the force control requirements, the 10 tasks can be categorized into three classes: four constant-force operations, four threshold-force operations, and two tasks that do not require force feedback. These tasks cover typical industrial applications such as assembly and surface processing, and include commonly used industrial tools such as screwdrivers, hammers, brushes, and scrapers.

Fig. 6 shows the setting of the physical equipment and working space. A 6-DOF robot arm (JAKA Zu3) equipped with a six-axis force/torque sensor (JK-SE-II-200) and a Robotiq 2F-85 gripper is mounted on an aluminum alloy base. A tool board is placed behind the robot to store various tools, while the front table area serves as the task execution workspace. Two RealSense D435i cameras (Camera-1 and Camera-2) are installed to monitor the task area and tool board, providing real-time visual feedback. For each task, the robot autonomously selects the appropriate tool from the tool board, moves to the designated workspace and then completes the required manipulation task.

The VLM used in this study is GPT-4o from OpenAI, which supports multimodal input understanding and complex context reasoning. All algorithms were executed on a workstation equipped with an Intel Core i7–12700F CPU, 16 GB RAM, and an NVIDIA RTX 3080 GPU, with the entire system deployed on a Linux platform.

The experiments are organized as follows: Section 4.2 presents comparative and ablation tests on 10 representative tool manipulation tasks to evaluate the performance of the proposed ToolManip framework. In Section 4.3, we compare the proposed visual prompt design with several baseline visual prompt methods to demonstrate its effectiveness in enhancing keypoint reasoning and task execution. In Section 4.4, a detailed analysis is conducted to examine the impact of different visual prompt factors, including keypoint number, visual marker size, and visual marker color, on the VLM's keypoint inference accuracy and localization performance. In Section 4.5, we further conduct comparative studies, including ablation experiments on the multi-agent architecture with agent-specific memory, cross-VLM generalization evaluation, representative failure case analysis, and latency assessment to examine the practical applicability of ToolManip.

4.2. *Comparison of open-vocabulary tool manipulation performance*

We compare ToolManip with two representative open-vocabulary robot manipulation baselines: MOKA [41] and OmniManip [42]. **MOKA** employs a pretrained vision-language model (GPT-4 V) to select keypoints from candidate points along the object's contour through visual question answering. Then a grid is overlaid on the workspace image and prompts the VLM to determine waypoints, which are connected sequentially to generate the robot's trajectory. **OmniManip** defines an object-centric canonical space based on functional affordances, and generates keypoints and principal-axis-aligned directions within this space for VLM (GPT-4o) to perform spatial reasoning. To enhance robustness, it further integrates interaction rendering and primitive resampling for closed-loop refinement during execution. In contrast, our proposed **ToolManip** integrates multimodal constraints by combining fine-grained positional cues—keypoint and interaction direction candidates with explicit force constraints enforced through force-control primitives. Additionally, a step check mechanism is incorporated for

Algorithm 1
Motion planning and step checker.

```
Parameters: Step index i, Failure number j, Step Image $I_i$, Step prompted Image $I_{ri}$, Step error $E_i = \{\Delta T_i, \Delta F_i\}$, Robot State $X_i = \{T_i, F_i\}$
Input: Primitive list of each task step $P_i = \{\mathscr{M}_i, \mathscr{F}_i\}$, Success criterion $\phi_i$
Output: Task Success or Failed
1: i ← 0, j ← 0, maxSteps ← $N_1$, maxTry ← $N_2$, StepResult←Success
2: while $i \le N_1$ do
3:   $i$ = Initialize(StepResult, i)
4:   ExecuteAction ($\mathscr{M}_i, \mathscr{F}_i$)
5:   $I_{ri}$ = VisualPrompt($I_i$), $E_i$ = ErrorCalculate ($X_{expect}, X_i$)
6:   StepResult = Inspector.Check($I_{ri}$, $\phi_i$)
7:   if StepResult = "Failed" then
8:     j ←$j + 1$, $P_{i-1}, P_i$ ← Inspector.Refine($I_{ri}$, $\phi_i$)
9:     if $j > N_2$ then
10:        return Task Failed
11:     end if
12:   else if StepResult = "Refine" then
13:     Update $P_i \leftarrow P_i + E_i$
14:   else if StepResult = "Success" then
15:     return StepResult
16:   end if
17: end while
18: return Task Success
```

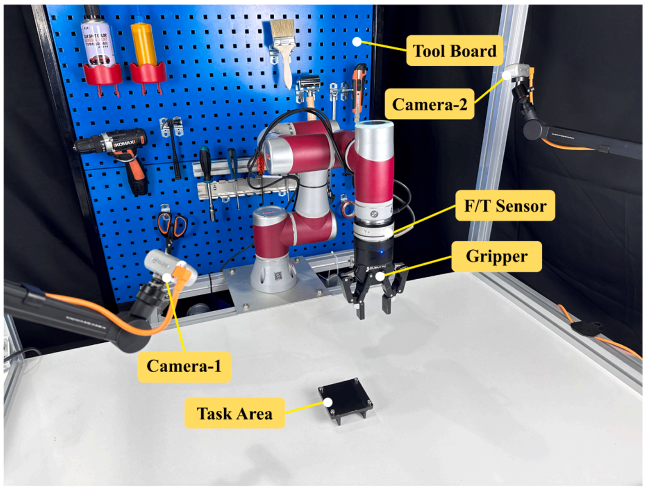


Fig. 6. Physical experiment settings for tool manipulation tasks.

closed-loop monitoring and adaptive adjustment. To ensure a fair comparison and exclude differences arising from varying VLMs capabilities, all methods adopt GPT-4o as a unified vision-language model.

Fig. 7 shows an example of the method comparison in the putty removal task. We can see that both ToolManip and OmniManip achieve high positioning accuracy for the keypoints and interaction directions, while MOKA shows obvious misalignment, leading to deviations from the intended scraping path. Additionally, the final surface results reveal that both MOKA and OmniManip tend to remove putty effectively at the start but leave residues toward the end of the motion. In comparison, ToolManip achieves more uniform and thorough removal along the entire trajectory. This improvement is likely attributed to the integration of force control primitives in ToolManip, which maintains consistent contact pressure throughout the task and helps ensure stable, complete execution.

All methods were evaluated on ten tool operation tasks with each task repeated 20 times. The overall task performance results of each method are summarized in Table 2. It can be seen that ToolManip consistently achieved the highest success rate across all tasks, maintaining over 90 % in every scenario. Notably, it attained 100 % success on the roller cleaning, valve switching, button pressing, and item sorting tasks. For tasks demanding precise constant force control, ToolManip significantly outperformed both baselines. This is likely because successful execution of these tasks depends on maintaining stable contact and continuous force interaction between the tool and the target, while MOKA and OmniManip rely solely on kinematic constraints and do not incorporate force control mechanisms. As a result, contact forces during operating process can induce relative positional shifts between the tool and the target, compromising task stability and reducing success rates. In contrast, ToolManip incorporates constant-force primitives that ensure reliable force output and maintain continuous kinematic alignment, thus guaranteeing robust tool–object interaction throughout the task and improves the task performance.

Additionally, ablation test results indicate that removing the Step Checker mechanism leads to a significant drop in success rates across all tasks as shown in the last column of Table 2. This is probably because uncorrected pose deviations caused by visual detection noise and force fluctuations during contact operations could compromise the accuracy of tool alignment and the stability of contact forces. By incorporating the Step Checker, such pose and force errors are detected and refined after each step, which helps maintain reliable interaction between tool and target, thereby improving overall task performance.

4.3. *Comparison of visual prompting with benchmark methods*

To examine the impact of different visual prompt designs on the VLM's keypoint reasoning accuracy and tool manipulation performance, we compare our visual prompting method with representative baselines and ablation methods. Fig. 8 illustrates the visual prompt designs of the comparison methods, highlighting the positional cue annotations on both the tool and target images. The comparative experiments are conducted on the putty removal task, which poses significant challenges in maintaining precise positional constraints and stable force control. Moreover, the task performance can be quantitatively evaluated by measuring the putty removal ratio. Three typical visual prompt methods and two ablation settings include:

- **MOKA:** it utilizes GroundedSAM [46] to extract the segmentation masks of target objects conditioned on text prompts. Keypoint candidate set is constructed by farthest point sampling along the mask contour and the geometric centroid.
- **OmniManip:** it adopts the SCAFFOLD [43] visual prompting method, which overlays a Cartesian grid of candidate points on the image with explicit coordinate annotation.
- **ReKep [40]:** it uses DINOv2 [49] to extract image features and segments the target mask with SAM. Structured candidate keypoints are generated by feature clustering within the mask.

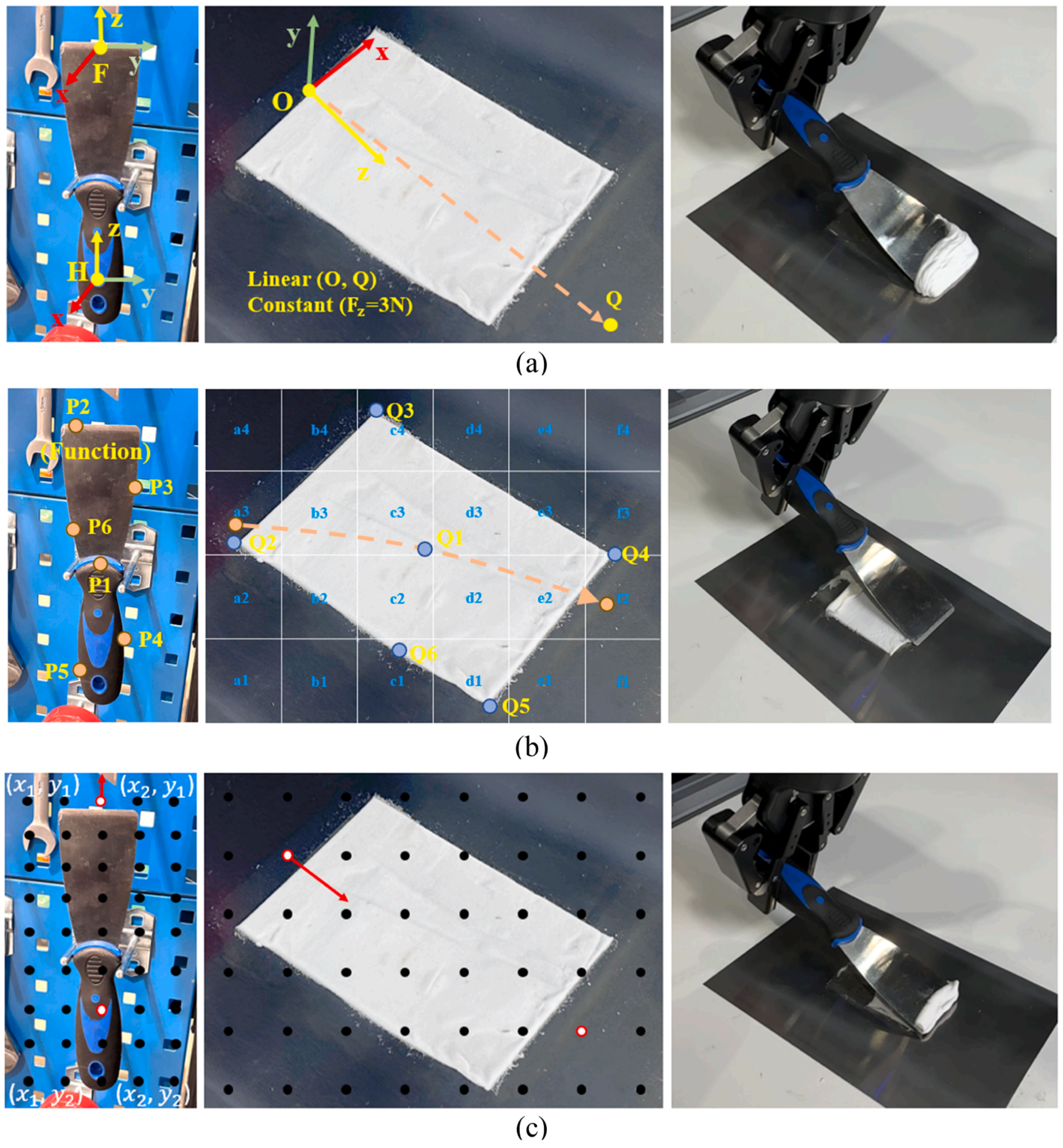


Fig. 7. Comparison with benchmark methods in the putty removal task. (a) ToolManip. (b) MOKA. (c) OmniManip. The first two columns show the visual prompts on the tool image and task scene image of different methods, and the third column shows the corresponding execution results.

- **ToolManip w/o HRE** (ablation setting): it applies LangSAM to segment the target region but skips the Hierarchical Region Extraction. Candidate keypoints are directly annotated on the full image without ROI cropping and scaling.
- **ToolManip w/o SIF** (ablation setting): it performs LangSAM segmentation and ROI cropping but omits the Structured Interaction Field. Candidate keypoints are randomly sampled within the mask area.

Since some baseline methods do not explicitly define directional constraints, we standardized the interaction directions across all methods by manually specifying them. Furthermore, to ensure fair comparison and consistent low-level execution, all methods use the same motion primitives and force control primitives during task execution. For each method, the putty removal task was tested 30 times. Table 3 summarizes the comparison results. The keypoint localization error represents the Euclidean distance between the keypoints inferred by the VLM and the ideal keypoints manually defined by human annotation. The grasp success is defined as a stable tool grasp without pose deviation and the task success is defined as achieving a putty removal ratio of at least 80 %. The results show that our method achieves the highest grasp and task success rates, with the lowest localization errors of all keypoints, demonstrating the effectiveness of proposed visual prompting mechanism in enhancing VLM inference precision and task feasibility.

From Table 3, we can also see that without Hierarchical Region Extraction (HRE), the keypoint localization errors increase significantly and task performance deteriorates. This probably results from that when the VLM processes a full scene image, it must implicitly locate the relevant task region while reasoning over the candidate keypoints within distracting background content, which disperses attention and reduces spatial inference accuracy. In contrast, providing a focused ROI restricts the context to task-relevant features, enabling more precise and consistent keypoint reasoning. While without Structured Interaction

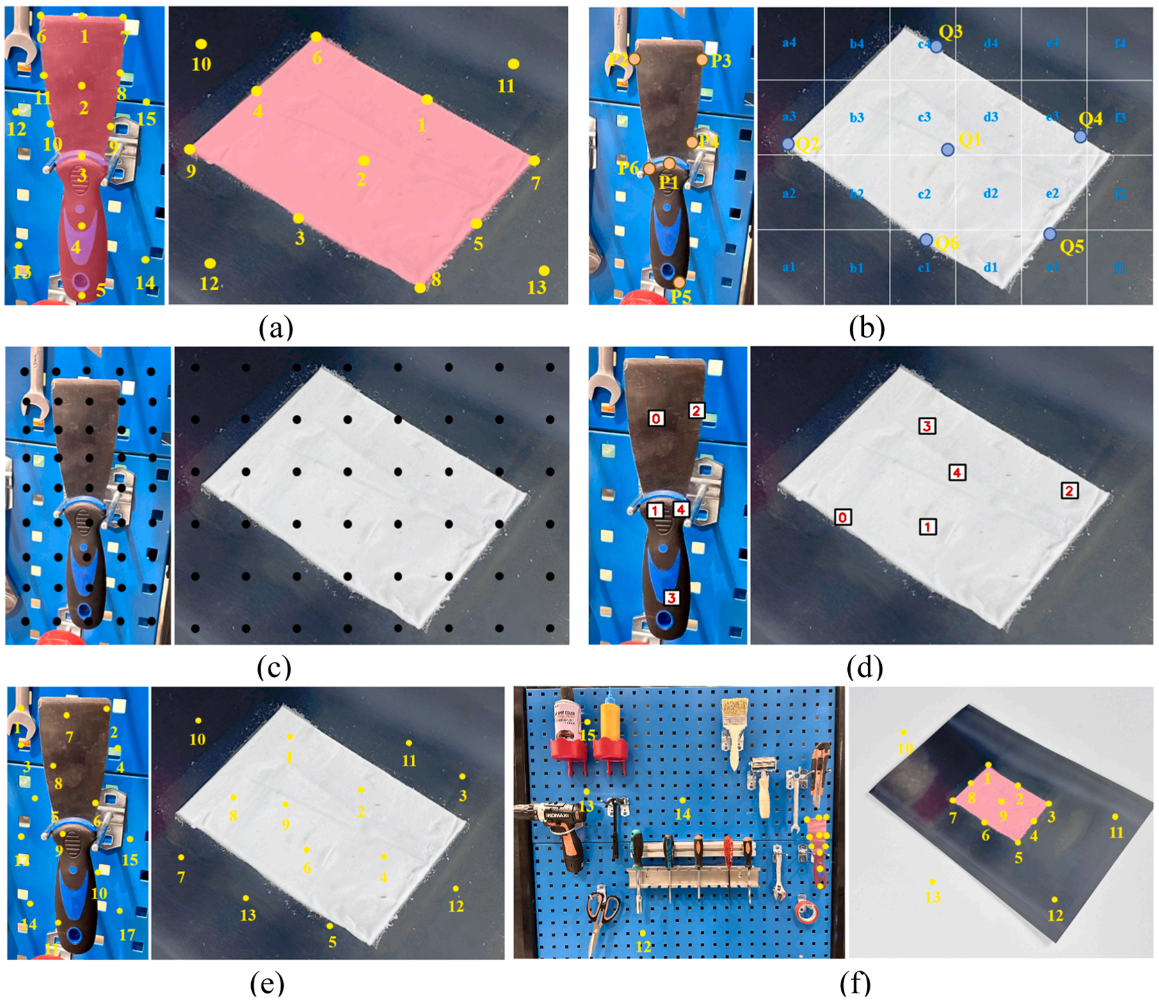
Table 2
Task success rate of different methods. The bolded numbers represent the best performance.

Force Requirement	Tasks	MOKA	OmniManip	ToolManip (ours)	
				w/t step checker	w/o step checker
Constant	Putty Removal	0.30	0.40	**0.90**	0.75
	Board Cutting	0.35	0.50	**0.95**	0.80
	Roller Cleaning	0.35	0.60	**1.00**	0.85
	Valve Switching	0.55	0.60	**1.00**	0.70
Threshold	Nail Hammering	0.80	0.85	**0.95**	0.75
	Button Pressing	0.70	0.80	**1.00**	0.85
	Bolt Tightening	0.80	0.90	**0.95**	0.75
	Screw Loosening	0.85	**0.90**	**0.90**	0.70
Off	Item Sorting	0.90	**1.00**	**1.00**	0.90
	Bearing Lubrication	0.85	0.80	**0.95**	0.80

Field (SIF), the candidate keypoints are randomly scattered over the image, which often fails to adequately cover critical geometric features such as edges, corners, and axial lines. This unstructured placement reduces the chance of selecting keypoints that truly reflect the target's physical and functional features, thereby decreasing the accuracy of keypoint reasoning and the stability of task execution. In contrast, by integrating both HRE and SIF, our method combines focused ROI

Table 3
Comparison of the visual prompt performance on the putty removal task among different methods. The bolded numbers represent the best performance.

Method	Keypoint Localization Error (mm)				Grasp Success Rate	Task Success Rate
	Grasp Point	Function Point	Start Point	End Point		
ToolManip (ours)	**6.7 ± 2.1**	**7.3 ± 1.8**	**6.9 ± 2.4**	**7.8 ± 2.6**	**29/30**	**27/30**
MOKA	13.4 ± 4.5	11.0 ± 4.1	11.7 ± 1.3	12.9 ± 1.5	25/30	24/30
OmniManip	9.8 ± 3.5	8.4 ± 3.7	9.1 ± 4.1	8.3 ± 4.5	27/30	25/30
ReKep	18.8 ± 6.8	16.9 ± 6.3	20.3 ± 7.1	22.5 ± 7.4	23/30	18/30
ToolManip w/o HRE	67.5 ± 19.8	70.2 ± 21.2	18.9 ± 5.5	20.4 ± 6.1	14/30	9/30
ToolManip w/o SIF	27.1 ± 7.1	25.5 ± 6.9	26.4 ± 6.5	29.1 ± 7.7	21/30	17/30



. **8.** Comparison of visual prompt configurations for different methods. (a) ToolManip. (b) MOKA. (c) OmniManip. (d) Rekep. (e) ToolManip w/o SIF. (f) ToolManip w/o HRE.

extraction with structured candidate keypoint placement. This ensures more consistent and complete geometric coverage, which significantly improves inference reliability and task performance.

We also compare the average putty removal ratio for each method, as presented in Fig. 9. Our approach achieves the highest average removal ratio of 88.1 %, whereas removing HRE or SIF causes a substantial drop to 33.6 % and 60.1 %, respectively. These results further demonstrate the critical contribution of both the HRE and SIF mechanisms to practical manipulation effectiveness.

4.4. Evaluation of visual prompt factors' influence on keypoint reasoning

To systematically assess how visual prompt design factors influence the VLM's ability to localize keypoints, we performed a series of tests focusing on three critical factors: the number of keypoints, the size of the visual marker, and the color contrast of the visual marker. These tests aim to quantify the effects of different configurations on reasoning accuracy and to determine the optimal settings for achieving precise spatial inference. Two evaluation metrics are adopted:

- **Keypoint Match Rate:** the percentage of keypoints inferred by the VLM that match the ground truth keypoints manually selected from the same candidate set (the higher the better).
- **Mean Position Error:** the average Euclidean distance between the predicted keypoint locations and the manually annotated ground truth, indicating the spatial localization accuracy (lower values indicate higher precision).

The test configurations and results are summarized in Figs. 10–12 and Table 4.

(a) Keypoint Number

To evaluate how the number of candidate keypoints affects inference accuracy, three configurations were tested as shown in Fig. 10:

- **Low (8 points)**: 3 points along the central axis, 3 on the contour boundary, and 2 in background region.
- **Medium (15 points)**: 5 along the central axis, 6 on the boundary, and 4 in background region.
- **High (30 points):** 10 along the central axis, 12 on the boundary, and 8 in background region.

As shown in Table 4, both the low and medium configurations achieved high match rates. However, the low-density configuration resulted in a higher mean position error than the medium one. This is likely because an insufficient number of keypoints cannot fully represent the target's geometry, leading to higher localization errors. In contrast, an excessive number of keypoints introduces redundant candidates and noise, which degrades both the match rate and spatial stability. Overall, a moderate keypoint density offers a good balance between adequate geometric coverage and robustness to noise.

(b) Visual Marker Size

To examine the influence of marker size on keypoint detection accuracy, we set the area of circular keypoint marker to cover approximately 0.025 %, 0.1 %, and 0.4 % of the total image, respectively. The height of the numeric label was set equal to the diameter of the keypoint, as shown in Fig. 11. Experimental results show that excessively small visual markers significantly reduce the VLM's ability to detect and localize keypoints accurately, yielding the lowest match rate (57.5 %) and the highest position error. Although larger markers are visually more salient, they can overlap or obscure critical object features, diverting the model's attention and impairing accurate keypoint reasoning. In contrast, a medium-sized marker achieves the optimal trade-off, yielding the highest match rate (96.0 %) and the lowest localization error.

(c) Visual Marker Color

To investigate how the color contrast between visual markers and the background affects the VLM's keypoint reasoning accuracy, we designed experiments with different color contrast configurations. Specifically, four commonly used base colors (red, yellow, blue, and black) were selected. We calculated the perceptual color difference between the image and each base color in the CIELAB color space [50]. Based on the computed differences, the base colors were ranked to create three contrast levels: Low Contrast, Medium Contrast, and High Contrast. Additionally, a same color configuration was included as a baseline, as shown in Fig. 12.

The results indicate that high-contrast visual markers yield the best keypoint recognition performance, with the highest match rate (99 %) and the lowest mean position error (6.7 mm), whereas low-contrast markers significantly degrade performance, reducing the match rate to

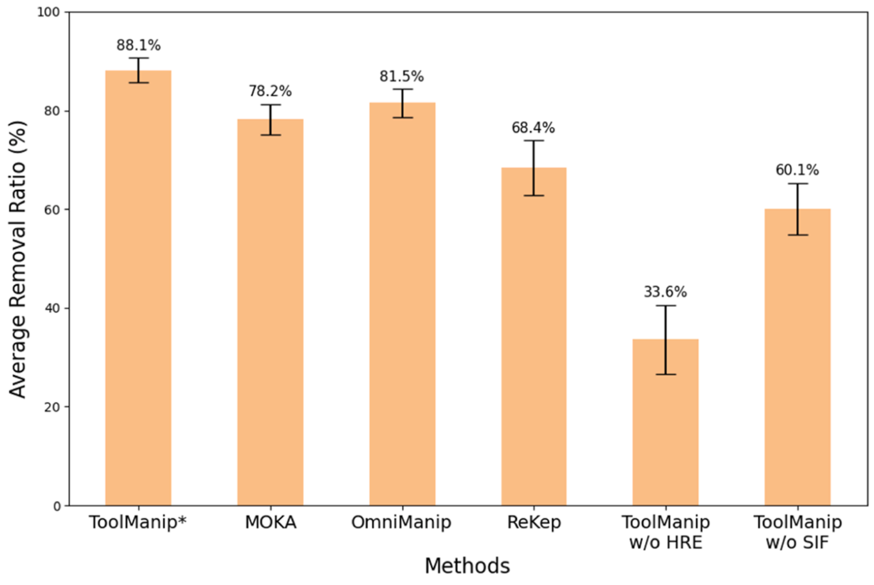


Fig. 9. Average putty removal ratio with standard error bars for comparison methods.

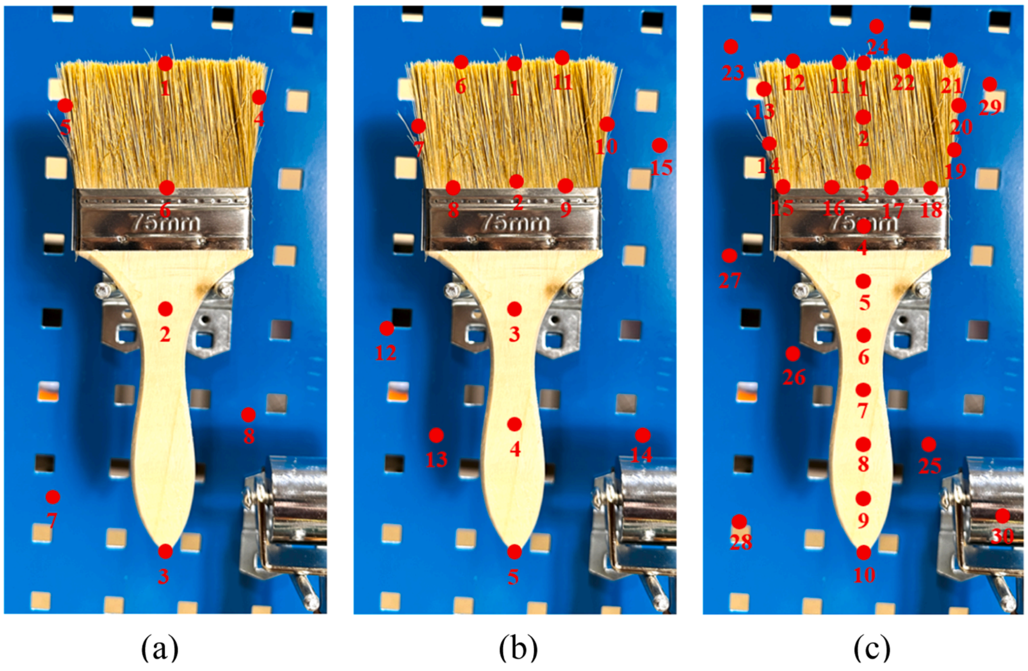


Fig. 10. Three settings of keypoint numbers for evaluating their impact on VLM keypoint prediction performance: (a) Low keypoint number; (b) Medium keypoint number; (c) High keypoint number.

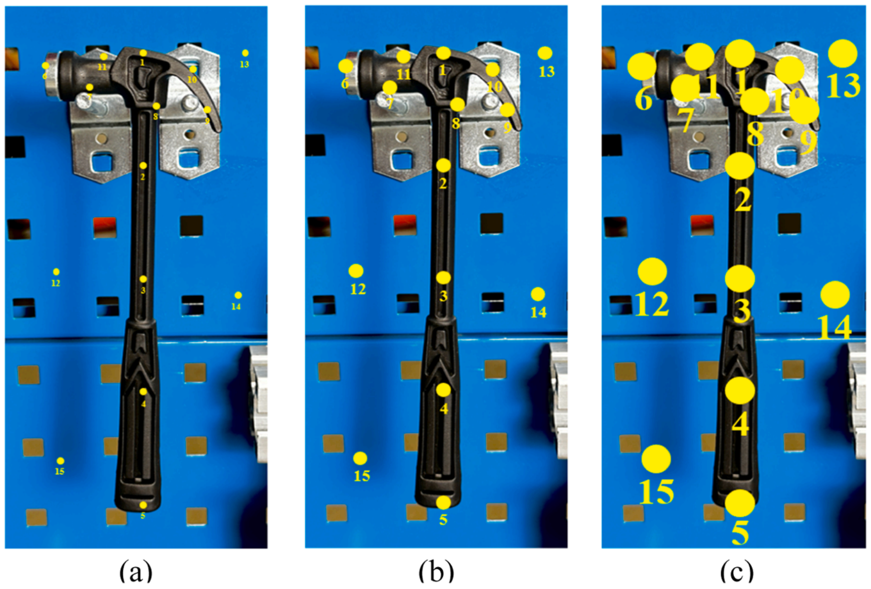


Fig. 11. Three settings of visual marker sizes for evaluating their impact on VLM keypoint prediction performance: (a) Small size; (b) Medium size; (c) Large size.

65 %. This improvement is likely due to that greater perceptual contrast enhances the saliency of the markers relative to the background, which helps the VLM focus on the relevant keypoint regions while suppressing interference from background textures. Consequently, clearer visual boundaries between the marker and the scene facilitate more precise spatial inference and contribute to robust downstream manipulation performance.

In summary, the results demonstrate that using a moderate number of well-distributed keypoints, suitably sized visual markers, and a high-contrast color scheme significantly improves the VLM's accuracy and reliability in keypoint inference. This leads to more precise keypoint localization and pose estimation, which in turn enhances the robustness and stability of robotic manipulation in practical tool manipulation tasks. Representative video recordings[2] are provided to illustrate the system's performance in different tool manipulation tasks.

4.5. *Comparison experiments to evaluate the proposed approach*

4.5.1. *Performance evaluation on the design of multi-agent architecture with agent-specific memory*

To validate the architecture design of our framework, we compared the performance of our framework in Item Sorting task with two ablated methods: (a) a single-agent baseline, where one VLM agent performs all

[2] See the videos of our experiments: https://youtu.be/qiC6lyETnOs

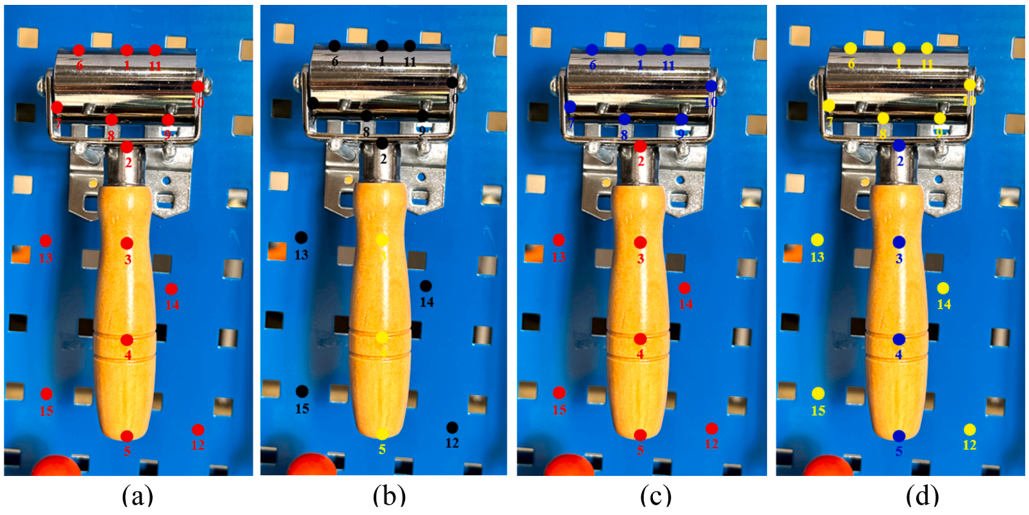


Fig. 12. Four settings of visual marker color contrast for evaluating their impact on VLM keypoint prediction performance: (a) Same color; (b) Low contrast; (c) Medium contrast; (d) High contrast.

Table 4
Impact of three visual prompt factors on the performance of VLM keypoint selection. Bolded numbers represent the best performance.

Factor	Setting	Keypoint Match Rate	Mean Position Error (mm)
Keypoint Number	Low	**99.0 %**	10.9 ± 1.9
	Medium	96.5 %	**7.3 ± 1.7**
	High	83.5 %	15.7 ± 4.3
Visual Marker Size	Small	57.5 %	24.8 ± 13.0
	Medium	**96.0 %**	**7.0 ± 1.5**
	High	87.5 %	10.4 ± 2.7
Visual Marker Color	Same	97.0 %	7.5 ± 1.8
	Low Contrast	65.0 %	18.1 ± 5.2
	Medium Contrast	93.5 %	8.9 ± 2.5
	High Contrast	**99.0 %**	**6.7 ± 1.2**

reasoning steps in a single prompt; and (b) a multi-agent variant without independent memory. As shown in Table 5, our full multi-agent architecture reduces the mean keypoint localization error to 8.3 ± 2.6 mm, a substantial improvement over the single-agent baseline (21.6 ± 13.5 mm). Moreover, it achieves 100 % tool selection and task success rates, compared to only 50 % and 30 %, respectively in the single-agent setting. These results demonstrate that explicit task decomposition is essential for accurate reasoning and robust task execution in complex scenarios.

In addition, without agent-specific memory, the multi-agent variant suffers from large variance in keypoint localization (9.8 ± 6.7 mm), reflecting unstable reasoning, and its task success rate drops to 90 %. With memory, our framework not only improves the mean accuracy (to 8.3 mm), but more importantly reduces the standard deviation from 6.7 mm to 2.6 mm, ensuring stable and consistent decision-making. These findings indicate that explicit task decomposition and agent-specific memory are critical for achieving accuracy, consistency, and robustness in complex industrial manipulation scenarios.

Table 5
Comparison of the performance on the Item Sorting task among different methods. The bolded numbers represent the best performance.

Method	Mean Keypoint Localization Error (mm)	Tool Selection Success Rate	Task Success Rate
ToolManip (ours)	**8.3 ± 2.6**	**1.00**	**1.00**
Single-agent variant	21.6 ± 13.5	0.50	0.30
ToolManip w/o memory	9.8 ± 6.7	**1.00**	0.90

4.5.2. *Cross-VLM generalization evaluation*

To further investigate the robustness and generalization capability of our framework, we evaluated ToolManip with different VLMs. In addition to the originally used GPT-4o, three representative VLMs released in 2025—Qwen2.5-VL, Gemma 3, and GPT-5—were integrated into our pipeline. Table 6 shows the experimental results on ten tool manipulation tasks. It can be observed that all models achieved success rates above 80 % across tasks, demonstrating the strong cross-model stability of our framework. Among them, GPT-5 achieved the highest performance in 9 out of 10 tasks, particularly excelling in constant-force and threshold-force scenarios, which require fine-grained multimodal reasoning. These results confirm that our framework exhibits low dependence on a specific VLM backbone and maintains high generalization and adaptability, enabling robust and precise tool manipulation across diverse models.

4.5.3. *Failure case analysis*

We also analyzed representative failure cases of ToolManip. As summarized in Table 7, errors were mainly observed in affordance localization (8.5 %) and robot execution (6.5 %). Here Occurrence Rate refers to the proportion of task steps in which a specific type of error was observed; Recovery Rate indicates the proportion of those errors that

Table 6
Task success rates of ToolManip equipped with different VLMs. The bolded numbers represent the best performance.

Force Requirement	Tasks	Qwen2.5-VL	Gemma 3	GPT-5	GPT-4o
Constant	Putty Removal	0.85	0.90	**0.95**	0.90
	Board Cutting	0.80	0.85	**0.95**	**0.95**
	Roller Cleaning	0.95	0.90	**1.00**	**1.00**
	Valve Switching	0.90	0.85	0.95	**1.00**
Threshold	Nail Hammering	0.90	0.95	**1.00**	0.95
	Button Pressing	**1.00**	0.90	**1.00**	**1.00**
	Bolt Tightening	0.90	**1.00**	**1.00**	0.95
	Screw Loosening	0.90	0.85	**0.95**	0.90
Off	Item Sorting	0.95	0.95	**1.00**	**1.00**
	Bearing Lubrication	0.85	0.90	**0.95**	**0.95**

Table 7
Analysis of Error Occurrence, Recovery, and Final Failure Rates.

Error Type	Occurrence Rate (%)	Recovery Rate (%)	Final Failure Rate (%)
Task Planning	2.5	2.5	0
Tool Selection	1.5	1.5	0
Affordance Localization	8.5	7.5	1
Primitive Reasoning	4	3	1
Robot Execution	6.5	4.5	2

were successfully corrected by the monitoring mechanism; and Final Failure Rate represents the percentage of errors that persisted and ultimately caused task failure. Fig. 13 shows that affordance localization errors typically involved incorrect keypoint or interaction direction prediction, leading to tool–object misalignment. Execution errors were primarily due to tool slippage from the gripper or unstable contact such as sliding during robot motion. Errors in primitive reasoning mainly arose from suboptimal force-torque settings, leading to incomplete or unstable execution in contact-intensive tasks. Most failures were successfully recovered by the monitoring mechanism, demonstrating that ToolManip can effectively detect and correct errors through its step-checking design.

4.5.4. Latency analysis

We conducted a latency analysis of ToolManip to quantify the computational overhead in both offline planning and online execution stages. As shown in Fig. 14, the offline planning process takes an average of about 60 s, with 75 % of the time consumed by VLM reasoning and the remainder by pose estimation, interaction field generation and result integration. In the online execution stage, each monitoring cycle takes around 8 s, primarily involving execution monitoring, pose/force estimation, and result integration. Although the current latency limits strict real-time use, it is acceptable for safety-critical, pre-plannable industrial tool operations, where precision and robustness are prioritized.

5. Conclusion

In this study, we propose ToolManip, a modular multi-agent Vision Language Model (VLM) framework for zero-shot robot tool manipulation in industrial environments. The framework decomposes manipulation process into four specialized modules—task understanding and planning, affordance reasoning, primitive reasoning, and execution monitoring—each integrated with a dedicated VLM agent with agent-specific memory, thereby enabling specialized reasoning across stages while maintaining consistency and stability throughout the manipulation process. To achieve higher accuracy in industrial tool manipulation, we introduce a visual prompting mechanism that provides precise task-relevant keypoints together with fine-grained interaction directions, and enhance primitive design by incorporating force regulation to support contact-intensive operations. Additionally, an integrated execution-monitoring pipeline further ensures reliable stepwise performance. Experimental results demonstrate that ToolManip achieves robust, generalizable, and high-accuracy manipulation across diverse, constraint-rich industrial tool manipulation tasks.

Our work empowers robots with reliable and generalizable tool-use capabilities to handle a wide range of complex, constraint-intensive industrial tasks, which can further enhance the intelligence and operational efficiency of smart manufacturing. This advancement in tool manipulation also contributes to the development of practical embodied intelligence systems, supporting their evolution towards versatile, adaptive, and precise operations in diverse real-world industrial environments.

One limitation of this work is the system sensitivity to camera viewpoint occlusions, which may affect the accuracy of VLM-based affordance reasoning. Future research will explore integrating multi-view visual inputs and enhanced spatial reasoning to improve geometric understanding and ensure consistent prediction of keypoints and interaction directions under varying viewpoints. Another limitation lies in the system latency, which restricts real-time applicability but remains acceptable for pre-plannable industrial tasks such as assembly and tool operation in fixed workstations. We also plan to develop per-agent error isolation mechanisms, including lightweight semantic–geometric consistency checks at module interfaces and an experience-based error library, to enable early detection and recovery from faulty VLM outputs.

Author statement

All authors acknowledge that the material presented in this manuscript has not been previously published, nor is it simultaneously under consideration by any other journal.

Qi Zhou (First Author): Led the design of the research pipeline, developed code, drafted manuscript, coordinated experiments.

Yuwei Gu: Designed and conducted real-world experiments, supported code developing.

Jiawen Li: Assisted with code developing and experimental testing.

Bohan Feng: Provided research resources and participated in data validation.

Boyan Li: Assisted in code development and experiment optimization.

Youyi Bi (Corresponding Author): Supervised the research project and managed the review and editing of the manuscript.

CRediT authorship contribution statement

Qi Zhou: Writing – original draft, Methodology, Conceptualization. **Yuwei Gu:** Validation, Software, Methodology. **Jiawen Li:** Software, Methodology. **Bohan Feng:** Validation, Software. **Boyan Li:** Software,

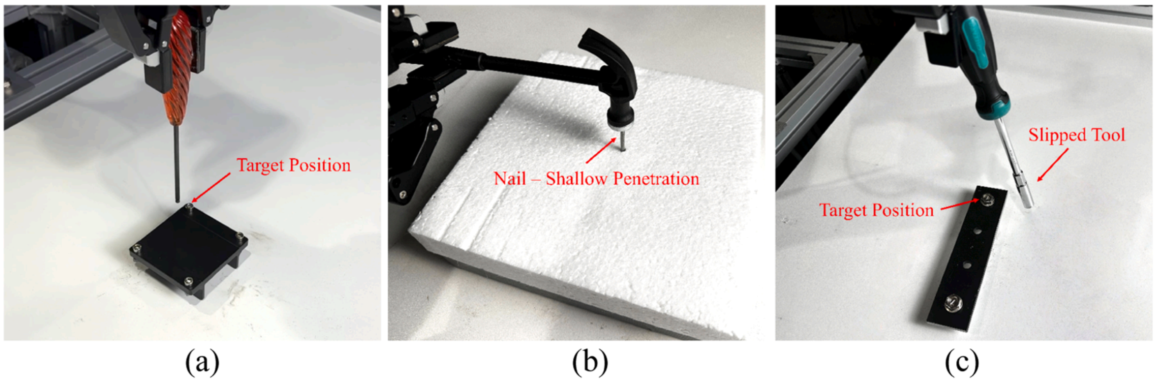


Fig. 13. Representative failure cases: (a) Keypoint localization error. (b) Insufficient applied force in threshold-force tasks. (c) Tool slippage during execution.

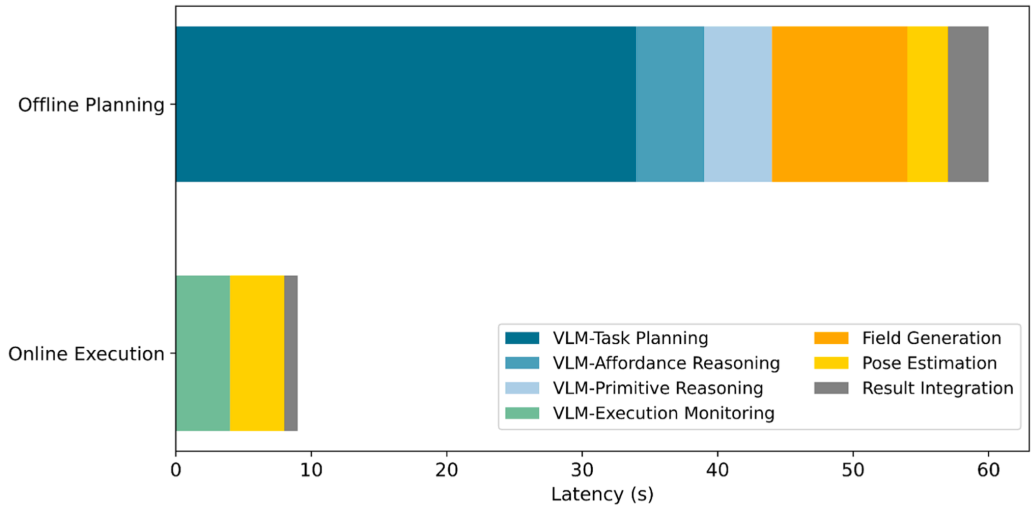


Fig. 14. Latency breakdown of ToolManip across offline and online stages.

Data curation. **Youyi Bi:** Writing – review & editing, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to acknowledge the financial support from the National Natural Science Foundation of China (52575300) and the National Key R&D Program of China (2022YFB4702400).

Appendix

Table A1
Prompt example for Task Planning.

Round 1: Identify the target object of {task} based on the task image. The required output format is: ⟨Object Name⟩, ⟨Position⟩, ⟨Color⟩. Round 2: List all visible tools on the tool board based on the tool image. Round 3: Select the best tool from {toolList} for a robot to accomplish {task}. The required output format is: ⟨Tool Name⟩, ⟨Position⟩, ⟨Color⟩. Round 4: Describe the task steps and success criterion for a robot with a two-finger gripper to finish {task} using {tool}. The required output format is: ⟨Step X: Subject, Action, Object, Success Criterion⟩.

Table A2
Prompt example for Affordance Reasoning.

The robot needs to finish {task}, the task steps involve: {taskStep}. Given images showing selected tool {tool} and target object {object} with annotated interaction keypoints and direction vectors, choose appropriate keypoints and vectors as the spatial positions to finish task.On the tool image, choose: (1) Keypoint H: Grasping point for robot gripper. (2) Keypoint F: Functional point located at tool's working area that will directly engage with the target object to perform the task. (3) Vector t: The operational direction vector at Keypoint F, representing the intended motion or force application direction of the tool during task execution. On the target object image, choose: (1) Keypoint O: The start contact point on the target object where the functional point of the tool will be aligned to initiate the operation. (2) Keypoint Q: The end point on the target object defining where the tool's operation will stop. (3) Vector o: The operational direction vector at Keypoint O, representing the intended motion or force direction along the target surface during the task. The required output format is: ⟨Keypoint H⟩, ⟨Keypoint F⟩, ⟨Vector t⟩, ⟨Keypoint O⟩, ⟨Keypoint Q⟩, ⟨Vector o⟩."

Table A3
Prompt example for Motion Planning.

Plan motion and force-control primitives for a robot with a two-finger gripper to accomplish {task}, where the operation is defined by a sequence of steps {taskStep}. Two annotated images are provided, depicting the tool and the scene with keypoints. From the primitive library {primitiveLibrary}, select appropriate motion and force primitives as the action commands for each task step. Positional parameters must be chosen from the annotated keypoints, and force-control parameters must follow the specifications in the force-control library {parameterLibrary}. The required output format is: StepX: ⟨Motion primitive⟩, ⟨Force primitive⟩.

Table A4
Prompt example for Execution Monitoring.

The robot is conducting task: {task}, current step: {step}, success criteria: {successCriteria}, the provided image shows the current state. The position and force/torque error of current state are: Pos error={posError}, force error={forceError}, torque error={torqueError}. Executed primitive: {motionPrimitive}, {forcePrimitive}. Based on this information, decide the execution status:
(1) Success: State satisfies success criterion, proceed to the next step.
(2) Improvable: Pose/Force deviation within correctable range, incrementally adjust primitive parameters.
(3) Failed: Unrecoverable error (e.g., tool drop, major misalignment), replan primitives for current and previous steps.
The required output format is: ⟨State Result⟩, ⟨Corrections⟩.

Data availability

Data will be made available on request.

References

[1] J. Zhang, H. Zhao, K. Chen, G. Fei, X. Li, Y. Wang, Z. Yang, S. Zheng, S. Liu, H. Ding, Dexterous hand towards intelligent manufacturing: a review of technologies, trends, and potential applications, Robot. Comput.-Integr. Manuf. 95 (2025) 103021, https://doi.org/10.1016/j.rcim.2025.103021.

[2] Í. Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, N. Arana-Arexolaleiba, A review on reinforcement learning for contact-rich robotic manipulation tasks, Robot. Comput.-Integr. Manuf. 81 (2023) 102517, https://doi.org/10.1016/j.rcim.2022.102517.

[3] Yang, F., Chen, W., Yang, K., Lin, H., Luo, D., Tang, C., Li, Z., and Wang, Y., 2024, "Learning granularity-aware affordances from Human-object interaction for tool-based functional grasping in dexterous robotics." https://doi.org/10.48550/arXiv.2407.00614.

[4] Z. Zhou, X. Yang, X. Zhang, Variable impedance control on contact-rich manipulation of a collaborative industrial mobile manipulator: an imitation learning approach, Robot. Comput.-Integr. Manuf. 92 (2025) 102896, https://doi.org/10.1016/j.rcim.2024.102896.

[5] D. Seita, Y. Wang, S.J. Shetty, E.Y. Li, Z. Erickson, D. Held, ToolFlowNet: robotic manipulation with tools via predicting tool flow from point clouds, in: Proceedings of The 6th Conference on Robot Learning, PMLR, 2023, pp. 1038–1049, in: https://proceedings.mlr.press/v205/seita23a.html.

[6] T. Fitzgerald, A. Goel, A. Thomaz, Modeling and learning constraints for creative tool use, Front. Robot. AI 8 (2021) 674292, https://doi.org/10.3389/frobt.2021.674292.

[7] Z. Liu, S. Tian, M. Guo, K. Liu, J. Wu, Learning to design and use tools for robotic manipulation, in: Proceedings of The 7th Conference on Robot Learning, PMLR, 2023, pp. 887–905, in: https://proceedings.mlr.press/v229/liu23b.html.

[8] Y. Zhu, P. Stone, Y. Zhu, Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation, IEEE Robot. Autom. Lett. 7 (2) (2022) 4126–4133, https://doi.org/10.1109/LRA.2022.3146589.

[9] Yang, Z., Garrett, C., Fox, D., Lozano-Pérez, T., and Kaelbling, L.P., 2024, "Guiding long-horizon task and motion planning with vision language models." https://doi.org/10.48550/arXiv.2410.02193.

[10] W. CAI, Y. MORI, N. SHIMADA, Generating robot action sequences: an efficient vision-language models with visual prompts, in: 2024 International Workshop on Intelligent Systems (IWIS), 2024, pp. 1–4, https://doi.org/10.1109/IWIS62722.2024.10706068.

[11] S. Li, Z. Yan, Z. Wang, Y. Gao, VLM-MSGraph: vision language model-enabled multi-hierarchical scene graph for robotic assembly, Robot. Comput.-Integr. Manuf. 94 (2025) 102978, https://doi.org/10.1016/j.rcim.2025.102978.

[12] Manuelli, L., Gao, W., Florence, P., and Tedrake, R., 2019, "kPAM: keyPoint affordances for category-level robotic manipulation." https://doi.org/10.48550/arXiv.1903.06684.

[13] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, S. Savarese, KETO: learning keypoint representations for tool manipulation, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 7278–7285, https://doi.org/10.1109/ICRA40945.2020.9196971.

[14] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, et al., PIVOT: iterative visual prompting elicits actionable knowledge for VLMs, in: Proceedings of the 41st International Conference on Machine Learning, PMLR, 2024, pp. 37321–37341, in: https://proceedings.mlr.press/v235/nasiriany24a.html.

[15] Xu, M., Huang, P., Yu, W., Liu, S., Zhang, X., Niu, Y., Zhang, T., Xia, F., Tan, J., and Zhao, D., 2023, "Creative robot tool use with large language models." https://doi.org/10.48550/arXiv.2310.13065.

[16] S. Tuli, R. Bansal, R. Paul, Mausam, TOOLTANGO: common sense generalization in predicting sequential tool interactions for robot plan synthesis, J. Artif. Intell. Res. 75 (2022) 1595–1631, https://doi.org/10.1613/jair.1.13791.

[17] Tang, C., Xiao, A., Deng, Y., Hu, T., Dong, W., Zhang, H., Hsu, D., and Zhang, H., 2025, "FUNCTO: function-centric one-shot imitation learning for tool manipulation." https://doi.org/10.48550/arXiv.2502.11744.

[18] K.P. Tee, S. Cheong, J. Li, G. Ganesh, A framework for tool cognition in robots without prior tool learning or observation, Nat. Mach. Intell. 4 (6) (2022) 533–543, https://doi.org/10.1038/s42256-022-00500-9.

[19] M. Qin, J. Brawer, B. Scassellati, Rapidly learning generalizable and robot-agnostic tool-use skills for a wide range of tasks, Front. Robot. AI 8 (2021) 726463, https://doi.org/10.3389/frobt.2021.726463.

[20] Z. Zhang, Z. Jiao, W. Wang, Y. Zhu, S.-C. Zhu, H. Liu, Understanding physical effects for effective tool-use, IEEE Robot. Autom. Lett. 7 (4) (2022) 9469–9476, https://doi.org/10.1109/LRA.2022.3191793.

[21] Z. Lu, N. Wang, C. Yang, A dynamic movement primitives-based tool use skill learning and transfer framework for robot manipulation, IEEE Trans. Autom. Sci. Eng. 22 (2025) 1748–1763, https://doi.org/10.1109/TASE.2024.3370139.

[22] H. Shi, H. Xu, S. Clarke, Y. Li, J. Wu, RoboCook: long-horizon Elasto-plastic object manipulation with diverse tools, in: Proceedings of The 7th Conference on Robot Learning, PMLR, 2023, pp. 642–660, in: https://proceedings.mlr.press/v229/shi23a.html.

[23] A. Hiranaka, M. Hwang, S. Lee, C. Wang, L. Fei-Fei, J. Wu, R. Zhang, Primitive skill-based robot learning from Human evaluative feedback, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Detroit, MI, USA, 2023, pp. 7817–7824, https://doi.org/10.1109/IROS55552.2023.10341912.

[24] Yin, Z.-H., Wang, C., Pineda, L., Hogan, F., Bodduluri, K., Sharma, A., Lancaster, P., et al., 2025, "DexterityGen: foundation controller for unprecedented dexterity." https://doi.org/10.48550/arXiv.2502.04307.

[25] Car, A., Yarlagadda, S.S., Bartsch, A., George, A., and Farimani, A.B., 2024, "PLATO: planning with LLMs and affordances for tool manipulation." https://doi.org/10.48550/arXiv.2409.11580.

[26] Qu, D., Song, H., Chen, Q., Yao, Y., Ye, X., Ding, Y., Wang, Z., et al., 2025, "SpatialVLA: exploring spatial representations for visual-language-action model." https://doi.org/10.48550/arXiv.2501.15830.

[27] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, D. Sadigh, Physically grounded vision-language models for robotic manipulation, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), 2024, pp. 12462–12469, https://doi.org/10.1109/ICRA57147.2024.10610090.

[28] Wang, B., Zhang, J., Dong, S., Fang, I., and Feng, C., 2024, "VLM see, robot do: human demo video to robot action plan via vision language model." https://doi.org/10.48550/arXiv.2410.08792.

[29] Dalal, M., Chiruvolu, T., Chaplot, D., and Salakhutdinov, R., 2024, "Plan-Seq-learn: language model guided RL for solving long horizon robotics tasks." https://doi.org/10.48550/arXiv.2405.01534.

[30] Guran, N.B., Ren, H., Deng, J., and Xie, X., 2025, "Task-oriented robotic manipulation with vision language models." https://doi.org/10.48550/arXiv.2410.15863.

[31] Joglekar, O., Lancewicki, T., Kozlovsky, S., Tchuiev, V., Feldman, Z., and Castro, D. D., 2024, "Towards natural language-driven assembly using foundation models." https://doi.org/10.48550/arXiv.2406.16093.

[32] Y. Lai, S. Yuan, Y. Nassar, M. Fan, A. Gopal, A. Yorita, N. Kubota, M. Rätsch, Natural multimodal fusion-based Human-robot interaction: application with voice and deictic posture via large language model, IEEE Robot. Autom. Mag. (2025) 2–11, https://doi.org/10.1109/MRA.2025.3543957.

[33] H. Fan, X. Liu, J.Y.H. Fuh, W.F. Lu, B. Li, Embodied intelligence in manufacturing: leveraging large language models for autonomous industrial robotics, J. Intell. Manuf. 36 (2) (2025) 1141–1157, https://doi.org/10.1007/s10845-023-02294-y.

[34] A. Mei, G.-N. Zhu, H. Zhang, Z. Gan, ReplanVLM: replanning robotic tasks with visual language models, IEEE Robot. Autom. Lett. 9 (11) (2024) 10201–10208, https://doi.org/10.1109/LRA.2024.3471457.

[35] Duan, J., Yuan, W., Pumacay, W., Wang, Y.R., Ehsani, K., Fox, D., and Krishna, R., 2024, "Manipulate-anything: automating real-world robots using vision-language models." https://doi.org/10.48550/arXiv.2406.18915.

[36] S.S. Kannan, V.L.N. Venkatesh, B.-C. Min, SMART-LLM: smart multi-agent robot task planning using large language models, in: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2024, pp. 12140–12147, https://doi.org/10.1109/IROS58592.2024.10802322.

[37] Z. Wang, R. Shen, B.C. Stadie, Wonderful team: zero-shot physical task planning with visual LLMs, Trans. Mach. Learn. Res. (2024). https://openreview.net/forum?id=udVkqIDYSM.

[38] M. Sharma, O. Kroemer, Generalizing object-centric task-axes controllers using keypoints, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 7548–7554, https://doi.org/10.1109/ICRA48506.2021.9561577.

[39] Ju, Y., Hu, K., Zhang, G., Zhang, G., Jiang, M., and Xu, H., 2024, "Robo-ABC: affordance generalization beyond categories via semantic correspondence for robot manipulation." https://doi.org/10.48550/arXiv.2401.07487.

[40] Huang, W., Wang, C., Li, Y., Zhang, R., and Fei-Fei, L., 2024, "ReKep: spatio-temporal reasoning of relational keypoint constraints for robotic manipulation." https://doi.org/10.48550/arXiv.2409.01652.

[41] K. Fang, F. Liu, P. Abbeel, S. Levine, MOKA: open-world robotic manipulation through mark-based visual prompting, Robot.: Sci. Syst. XX, Robot.: Sci. Syst. Foun. (2024), https://doi.org/10.15607/RSS.2024.XX.062.

[42] Pan, M., Zhang, J., Wu, T., Zhao, Y., Gao, W., and Dong, H., 2025, "OmniManip: towards general robotic manipulation via object-centric interaction primitives as spatial constraints." https://doi.org/10.48550/arXiv.2501.03841.

[43] Lei, X., Yang, Z., Chen, X., Li, P., and Liu, Y., 2024, "Scaffolding coordinates to promote vision-language coordination in large multi-modal models." https://doi.org/10.48550/arXiv.2402.12058.

[44] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, et al., Segment anything, in: 2023 IEEE/CVF International Conference on Computer Vision (ICCV), 2023, pp. 3992–4003, https://doi.org/10.1109/ICCV51070.2023.00371.

[45] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, et al., Grounding DINO: marrying DINO with grounded pre-training for open-set object detection, in: A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, G. Varol (Eds.), Computer Vision – ECCV 2024, Springer Nature Switzerland, Cham, 2025, pp. 38–55, https://doi.org/10.1007/978-3-031-72970-6_3, eds.

[46] Ren, T., Liu, S., Zeng, A., Lin, J., Li, K., Cao, H., Chen, J., Huang, X., Chen, Y., Yan, F., Zeng, Z., Zhang, H., Li, F., Yang, J., Li, H., Jiang, Q., and Zhang, L., 2024, "Grounded SAM: assembling open-world models for diverse visual tasks." https://doi.org/10.48550/arXiv.2401.14159.

[47] J. Lin, L. Liu, D. Lu, K. Jia, SAM-6D: segment anything model meets zero-shot 6D object pose estimation, in: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024, pp. 27906–27916, https://doi.org/10.1109/CVPR52733.2024.02636.

[48] L. Medeiros, Lang-segment-anything, GitHub [Online]. Available: https://github.com/luca-medeiros/lang-segment-anything/tree/main/, 2023 [Accessed: 23- Apr-2025].

[49] Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., et al., 2024, "DINOv2: learning robust visual features without supervision." https://doi.org/10.48550/arXiv.2304.07193.

[50] D. Durmus, CIELAB color space boundaries under theoretical spectra and 99 test color samples, Color Res. Appl. 45 (5) (2020) 796–802, https://doi.org/10.1002/col.22521.