

Article | Received 3 February 2026; Revised 7 May 2026; Accepted 13 May 2026; Published 22 May 2026
<https://doi.org/10.55092/rl20260018>

Robotic assembly via self-prompt Segment Anything Model and discrete prompt optimization



Qi Guo, Xing Liu*, Haitao Chang, Zhengxiong Liu and Panfeng Huang

Research Center for Intelligent Robotics, School of Astronautics, Northwestern Polytechnical University, Xi'an, China

* Correspondence author; E-mail: xingliu@nwpu.edu.cn.

Highlights:

- This paper introduces a self-prompt SAM that automatically generates structured visual representations of assembly parts from raw images.
- We propose a discrete prompt optimization mechanism that iteratively refines black-box MLLM prompts, boosting recognition stability to 95% and reducing reasoning attempts by 48%.
- Our method is experimentally validated on two robotic assembly tasks, successfully generating and executing feasible assembly plans directly from visual inputs.

Abstract: In complex assembly scenarios, Multimodal Large Language Models (MLLMs), despite their strong vision-language understanding capabilities, remain limited in their ability to produce structured and executable assembly plans directly from raw visual observations. This difficulty is particularly evident in black-box settings, where prompt design depends heavily on human experience and repeated trial-and-error, often leading to unstable results and high iteration costs. To address these issues, this paper presents a Perception-Recognition-Planning-Action (PRPA) framework for robotic assembly that enables the direct derivation of assembly instructions from scene images. The framework incorporates two key components. A self-prompt Segment Anything Model (SAM) is used to automatically generate structured and verifiable visual representations of assembly parts, ensuring consistent inputs for subsequent reasoning. In addition, a discrete prompt optimization mechanism is introduced to refine prompts for black-box MLLMs through iterative quality assessment and targeted symbolic modifications, improving the reliability of part recognition, semantic attribute extraction, and functional relationship modeling. Together, these components allow the system to generate temporally ordered and physically feasible assembly action sequences, which are represented as symbolic assembly plans suitable for both human interpretation and robotic execution. By combining MLLM-based reasoning with structured assembly planning, the proposed approach shifts the role of language models from interpreting predefined instructions to directly supporting instruction generation from visual input. Experimental results show that the proposed prompt optimization mechanism reduces the average number of reasoning attempts by 48% and achieves 95% stability in part recognition.



Copyright©2026 by the authors. Published by ELSP. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium provided the original work is properly cited.

Keywords: assembly sequence planning; MLLM-based semantic understanding; self-prompt SAM; prompt optimization

1. Introduction

Long-horizon assembly manipulation remains a fundamental challenge in robotics [1,2]. The primary difficulty does not lie in executing individual actions, but in enabling a robot to perform coherent perception, semantic reasoning, and physically feasible planning across multiple interdependent components. In complex assembly scenarios, a robot must accurately identify parts from visual observations, understand their geometric and functional properties, and generate a temporally ordered sequence of operations under spatial, kinematic, and physical constraints. Each action must be executable at the local level while remaining consistent with the global task objective and overall physical feasibility of the assembly process.

Recent advances in Multimodal Large Language Models (MLLMs) have demonstrated strong capabilities in parsing visual scenes and reasoning over natural language instructions. By leveraging large-scale multimodal data, these models offer promising tools for high-level semantic understanding and task reasoning in robotics. However, when applied to assembly tasks, MLLMs often fail to produce stable, structured, and executable plans directly from raw visual inputs. While they may generate plausible textual descriptions or high-level instructions, such outputs frequently lack the structure and consistency required for reliable robotic execution, leading to an evident gap between semantic understanding and actionable planning. This executability gap primarily stems from two fundamental bottlenecks.

Bottleneck 1: Unstructured visual entry causing reasoning drift. Assembly tasks often involve scenes with numerous parts, complex geometries, cluttered backgrounds, and indistinct object boundaries. When part segmentation relies on manual annotation or error-prone detection pipelines, perceptual errors propagate to later stages, distorting semantic recognition and relational reasoning. This degradation stems from the lack of structured, consistent visual inputs, which prevents MLLMs from building stable representations of parts and their spatial and functional relations.

To address this, we redesign the perception stage as a self-prompt segmentation pipeline. Instead of depending on handcrafted prompts, the MLLM first analyzes the assembly image and autonomously generates bounding-box and point prompts for the Segment Anything Model (SAM). These prompts are iteratively refined through quality validation and feedback-driven re-segmentation, resulting in structured, semantically grounded visual representations. By converting raw images into stable visual cues, our approach reduces perceptual uncertainty and supplies reliable inputs for downstream reasoning and planning.

Bottleneck 2: Non-scalable prompt engineering leading to unstable semantic outputs. State-of-the-art MLLMs are generally closed-source systems accessible only via Application Programming Interfaces (APIs), ruling out gradient-based tuning and internal parameter adjustments [3,4]. Consequently, traditional prompt engineering relies heavily on manual effort and trial-and-error, which is costly and yields inconsistent responses—especially in assembly tasks where parts vary widely and instructions are often underspecified or ambiguous. While gradient-based prompt optimization methods exist [5,6], they demand internal model access, substantial fine-tuning, and large task-specific datasets, rendering them impractical for black-box MLLMs. Recent gradient-free techniques improve prompts through enumeration or scoring but often lack

iterative feedback and domain-aware refinement, resulting in biased or incomplete semantic comprehension.

To bridge this gap, we introduce a discrete prompt optimization module designed for black-box MLLMs. Here, prompts are treated as editable symbolic constructs rather than static templates. Through cyclic evaluation, semantic error localization, and targeted editing strategies, prompts are progressively refined to produce consistent, structured, and verifiable semantic outputs—all without accessing the model’s internal parameters. This enables robust part recognition, attribute extraction, and relational inference under black-box constraints.

2. Related work

2.1. Assembly Sequence Planning

Robotic assembly refers to the operation in which a robot manipulates a component to be inserted or joined with another. This type of operation is prevalent not only in traditional manufacturing assembly processes but also in a wide range of non-manufacturing applications, such as on-orbit assembly in space environments. Due to the presence of various uncertainties during robotic assembly, interactions such as contact and collision between components and the surrounding environment are inevitable. As a result, robotic assembly is typically characterized as a constrained motion task. Existing research on Assembly Sequence Planning (ASP) encompasses several paradigms, including graph-based methods [7], knowledge-based methods [8,9], intelligent optimization approaches [10–15], and artificial intelligence-driven techniques [16,17].

Sequence planning methods based on intelligent optimization primarily leverage heuristic and optimization algorithms. Heuristic approaches, such as Genetic Algorithms (GA) [10–12], have shown promise in solving small-scale ASP problems. However, their performance often deteriorates when applied to complex assemblies, as they are highly sensitive to the quality of the initial solution and are susceptible to premature convergence to local optima. Optimization algorithms, including Particle Swarm Optimization (PSO) [13] and Ant Colony Optimization (ACO) [14], mitigate local optimality to some extent by employing swarm intelligence strategies. Nonetheless, their overall efficiency and solution quality remain significantly constrained by their dependency on high-quality initial solutions.

2.2. Learning for Assembly Sequence Planning

Learning-based approaches hold great promise in addressing the combinatorial challenges inherent in assembly sequence generation. Self-supervised learning methods [18] infer assembly rules by reversing the disassembly process and leveraging learned priors on visual shapes and spatial alignment capabilities, with an emphasis on generalization in the absence of Computer Aided Design (CAD) models. Reinforcement learning and approximate dynamic programming have been employed to tackle the combinatorial explosion in assembly planning [19–21]. Among them, Watanabe *et al.* [19] integrates structural optimization, policy learning, and motion execution, exhibiting a strong hierarchical architecture. In [22], supervised learning is used to perform internal planning, where a Graph Neural Network is trained to predict disassembly sequences that are physically feasible for complex 3D components. Furthermore, Dalal *et al.* [23] demonstrates that motion planning can be utilized to track plans generated by large language models, enabling the effective acquisition of low-level robotic control skills without relying on predefined skill libraries.

Beyond traditional prompt tuning methods, recent advances in visual prompting and in-context learning have opened new avenues for robotics. Robotics Transformer 2 (RT-2) [24] demonstrates that vision-language-action models can directly output robotic actions by fine-tuning on large-scale robot data, enabling generalization to novel tasks. Pathways Language Model Embodied (PaLM-E) [25] incorporates embodied reasoning by conditioning on multimodal prompts that combine visual observations with textual instructions, generating robot action plans without task-specific fine-tuning. First, unlike RT-2 and PaLM-E that rely on large-scale robot data or model fine-tuning, our method operates under strict black-box constraints where internal model parameters are inaccessible. Second, while PromptCraft focuses on manual prompt engineering, we propose an automated, iterative discrete prompt optimization mechanism driven by quality feedback. Thus, our approach complements existing work by addressing scenarios where model access is limited and manual prompt design is impractical.

3. Methods

We propose a multi-stage integrated framework for robotic assembly tasks, which includes four core layers: Perception, Recognition, Planning, and Action, as shown in Figure 1. This framework utilizes MLLMs as its core inference component, combined with self-prompt SAM and a discrete prompt optimization loop, to achieve semantic understanding and functional relationship modeling of parts in the assembly scenario. As a result, executable operation instructions are generated to enable the full automation of the assembly task.

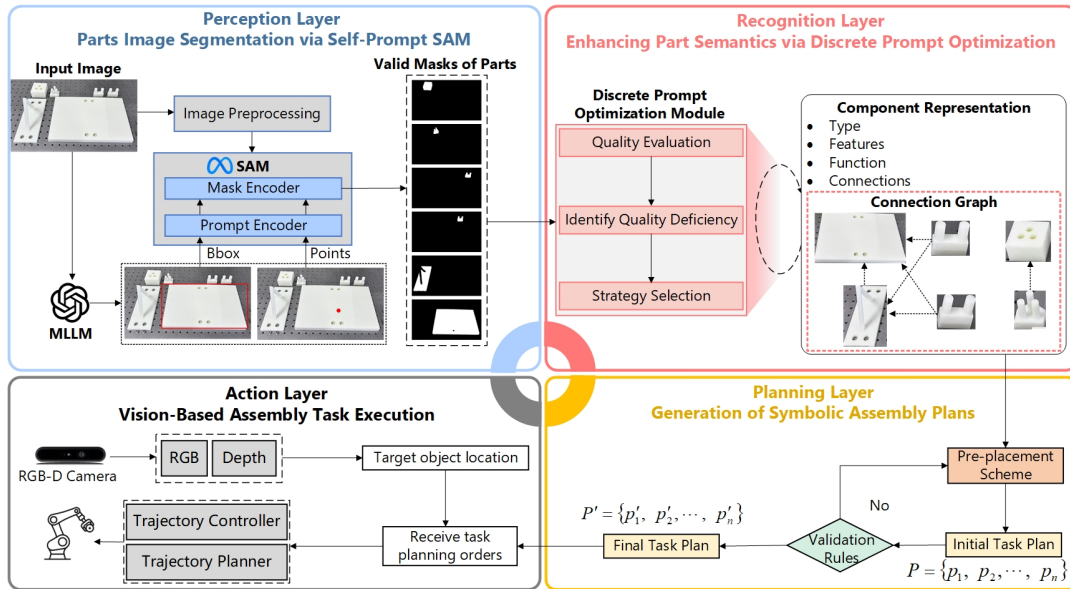


Figure 1. The overview architecture of the proposed PRPA illustrates the four core layers (Perception, Recognition, Planning, and Action) and their interactive workflows. The perception layer employs a self-prompt SAM for part segmentation and structured visual prompt generation; the recognition layer leverages MLLMs with adaptive prompt optimization to achieve semantic understanding of parts and their relationships; the planning layer converts structured semantic information into executable symbolic assembly plans; and the action layer executes assembly tasks through vision-based localization and motion control, forming a closed loop from perception to execution.

3.1. Parts image segmentation via self-prompt SAM: perception

The perception layer is designed as a modular processing pipeline centered around the SAM, comprising multiple collaborative functional agents that include image preprocessing, prompt generation, automatic segmentation execution, quality verification, image enhancement, and feature extraction. Unlike the standard SAM, which relies on manually annotated prompts (e.g., bounding boxes or points) provided by users, our self-prompt SAM automatically generates these prompts using an MLLM that analyzes the raw assembly image. Specifically, the MLLM identifies relevant parts and produces bounding-box and point prompts along with textual descriptions, which are then fed into SAM for segmentation. This eliminates the need for manual annotation and enables a fully automatic pipeline. The primary role of this pipeline is to generate high-quality structured visual prompts from raw component images, thereby supporting the reasoning capabilities of the subsequent MLLM in semantic understanding and relational modeling. Moreover, the built-in adaptive feedback mechanism ensures that the segmentation results are accurate, interpretable, and structurally consistent.

Image preprocessing. The image processing module first preprocesses the raw image. Operations include Gaussian/median filtering for denoising, histogram equalization for contrast enhancement, and geometric correction. These steps produce standardized input for the segmentation module.

MLLM-guided visual prompt generation. Subsequently, the MLLM is introduced to conduct an initial analysis of the entire image. By parsing the spatial layout and semantic cues of components within the scene, the MLLM automatically identifies all relevant parts and generates discrete visual prompts for each one, including bounding boxes (bboxes) and bbox center point coordinates. Each prompt is accompanied by a textual description (e.g., “a square structure in the upper-left corner”), which serves as an automatic source of point-based guidance for SAM, replacing traditional manual click operations. These prompts generated by the MLLM require no additional supervision, relying solely on the model’s understanding of the image. They are fed into the SAM framework, where they are encoded by the prompt encoder and fused with image embeddings extracted by the image encoder. This process produces instance-level segmentation masks for each component, enabling fine-grained part extraction without the need for manual annotation. To ensure segmentation completeness and boundary accuracy, a dual-metric quality verification mechanism is introduced based on Intersection over Union (IoU) and boundary error (BE).

$$\text{IoU}(M, G) = \frac{|M \cap G|}{|M \cup G|} = \frac{\sum_{i,j} M_{ij} \cdot G_{ij}}{\sum_{i,j} \max(M_{ij}, G_{ij})} \quad (1)$$

$$\text{BE}(M, G) = \max \left\{ \max_{p \in \partial M} d(p, \partial G), \max_{q \in \partial G} d(q, \partial M) \right\} \quad (2)$$

where $M \in \{0, 1\}^{H \times W}$ is the predicted binary segmentation mask by SAM; $G \in \{0, 1\}^{H \times W}$ is the ground truth mask; M_{ij} and G_{ij} denote the pixel values at location (i, j) in M and G , respectively. ∂M and ∂G denote the boundary points of M and G , respectively; $d(p, \partial G) = \min_{g \in \partial G} \|p - g\|_2$ is the Euclidean distance from point p to the closest point on ∂G ; $d(q, \partial M) = \min_{m \in \partial M} \|q - m\|_2$ is defined similarly. A higher IoU value approaching 1 indicates a higher segmentation accuracy.

Given threshold values τ_{IoU} and τ_{BE} :

$$\mathcal{Q}(M, G) = \begin{cases} 1, & \text{if } IoU(M, G) \geq \tau_{IoU} \text{ and } BE(M, G) \leq \tau_{BE} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

If the results fail to meet the quality standards, a feedback loop is triggered that automatically adjusts SAM parameters—such as mask thresholds, boundary smoothness, and prompt point density—and re-executes the segmentation until the verification criteria are met.

Feature extraction. Upon passing verification, the image enhancement module refines the mask regions through detail enhancement and edge sharpening techniques, using adaptive contrast adjustment and Sobel/Canny operators to improve structural clarity and feature visibility. Subsequently, the feature extraction module extracts three categories of attributes from the segmented regions: (1) geometric features such as area, aspect ratio, and contour descriptors; (2) structural features including hole count, positions, and dimensions; (3) assembly-related features like connectors, protrusions, and grooves. A shape-matching scoring mechanism is employed to assist in inferring potential assembly relationships. All extracted information is then organized into a structured visual prompt, which serves as standardized input for the MLLM in the subsequent recognition layer.

3.2. Enhancing part semantics via discrete prompt optimization: recognition

To enhance the accuracy of part semantic understanding in robotic assembly tasks, we propose a task-specific discrete prompt optimization method tailored for assembly scenarios. In contrast to traditional approaches that rely on static prompts for model querying, our method performs optimization within a discrete prompt space, where the prompt is represented as a modifiable sequence of textual components. Rather than applying continuous gradient-based updates as in soft prompt tuning, we iteratively revise the prompt through symbolic transformations—such as rephrasing, expanding, or reorganizing its structure—based on identified quality issues. This process is driven by a closed-loop mechanism of “Quality Evaluation $\rightarrow$ Identify Quality Deficiency $\rightarrow$ Strategy Selection”, enabling task-specific prompt refinement over multiple rounds. As a result, the recognition layer generates a structured understanding of the target parts, including their geometric and physical attributes, functional roles, and all possible connection relationships between parts. This enriched representation serves as a reliable and high-quality semantic foundation for subsequent assembly planning.

3.2.1. Quality evaluation

We propose a multi-dimensional quality evaluation method, which comprehensively assesses the structural integrity and semantic validity of MLLM-generated content, rather than merely verifying the presence of an answer. This method formalizes the evaluation process through five distinct dimensions, each capturing a critical aspect of output quality. To quantitatively represent overall quality, we define a composite score (*Score*), computed by the function “Evaluate Quality”. Each evaluation dimension is assigned a predefined weight reflecting its relative importance, and the final score is obtained as a weighted sum of

the individual dimension scores:

$$Score = \sum_{i=1}^5 \omega_i \cdot s_i \quad (4)$$

where each individual score $s_i \in [0, 1]$ is computed based on specific evaluation criteria, and $\omega_i \in [0, 1]$ denotes the corresponding weight, with $\sum \omega_i = 1$. The final score $Score \in [0, 1]$ reflects the overall quality of the response and is used to guide prompt optimization.

s_1 denotes the score for JSON format completeness, and is computed based on three binary indicators: (i) whether the JSON format contains required markers, (ii) whether the content is syntactically parsable, and (iii) whether the parsed structure is non-empty. I_i is an indicator function, taking a value of 1 if the corresponding criterion is satisfied, and 0 otherwise; that is, $I_i \in \{0, 1\}$. The final score is obtained by a weighted sum of these indicators:

$$s_1 = \alpha_1 \cdot I_1 + \alpha_2 \cdot I_2 + \alpha_3 \cdot I_3 \quad (5)$$

s_2 denotes the score for the validity of key fields. Let N_{present} be the number of fields that exist and are not empty, N_{required} be the total number of required fields, n_{invalid} denote the number of fields that exist but contain invalid content, and $\beta \in [0, 1]$ be a penalty weight. Then the field completeness rate is defined as: $r = \frac{N_{\text{present}}}{N_{\text{required}}}$. The validity score s_2 is computed as:

$$s_2 = \min(1, \max(0, r \cdot (1 - \beta \cdot n_{\text{invalid}}))) \quad (6)$$

s_3 denotes the score for confidence threshold check. Let $c \in [0, 1]$ be the confidence value, $c_{\min}$ be the lower threshold, and $c_{\max}$ be the upper threshold. Then the confidence evaluation function is defined as:

$$s_3(c) = \psi(c; c_{\min}, c_{\max}) \quad (7)$$

$$\psi(c) = \begin{cases} 0, & \text{if } c \text{ is missing,} \\ \eta, & \text{if } c \notin \Omega, \\ 1, & \text{if } c \geq c_{\max}, \\ \kappa_1 + \kappa_2 \cdot \frac{c - c_{\min}}{c_{\max} - c_{\min}}, & \text{if } c_{\min} \leq c < c_{\max}, \\ \kappa_3 \cdot \frac{c}{c_{\min}}, & \text{if } 0 \leq c < c_{\min}. \end{cases} \quad (8)$$

where Ω denotes the valid domain of the confidence value, defined as $[0, 1]$. The parameters $\kappa_1, \kappa_2, \kappa_3 \in (0, 1]$ are scaling coefficients corresponding to different confidence intervals, and $\eta \in (0, 1)$ is a penalty constant assigned to abnormal confidence values. In this paper, the parameters are set to $\kappa_1 = 0.7$, $\kappa_2 = 0.3$, $\kappa_3 = 0.5$, and $\eta = 0.3$.

s_4 denotes the score for richness and completeness of content description, which integrates three components: the number of features score s_f , the description length score s_d , and the function description score s_g . The overall score is defined as:

$$s_4 = \min(1, s_f + s_d + s_g) \quad (9)$$

s_5 denotes the score for semantic consistency check, computed using three binary indicators: $I_{\text{type-func}}$ (whether the type matches the function), $I_{\text{feat-type}}$ (whether features match the type), and $I_{\text{desc-cons}}$ (whether the description matches the type or function). Each $I \in \{0, 1\}$, where 1 denotes consistency and 0 denotes inconsistency. γ_i are the corresponding weights.

$$s_5 = \gamma_1 \cdot I_{\text{type-func}} + \gamma_2 \cdot I_{\text{feat-type}} + \gamma_3 \cdot I_{\text{desc-cons}} \quad (10)$$

3.2.2. Identify quality deficiency

Building upon the quality evaluation stage, we integrate a prompt optimization strategy matrix that links each of the five evaluation dimensions to a corresponding class of common response deficiencies. For each dimension, a targeted refinement strategy is defined to guide the prompt optimization process in subsequent iterations, as shown in Figure 2. These five categories are dynamically detected during each iteration using the “analyze” function.

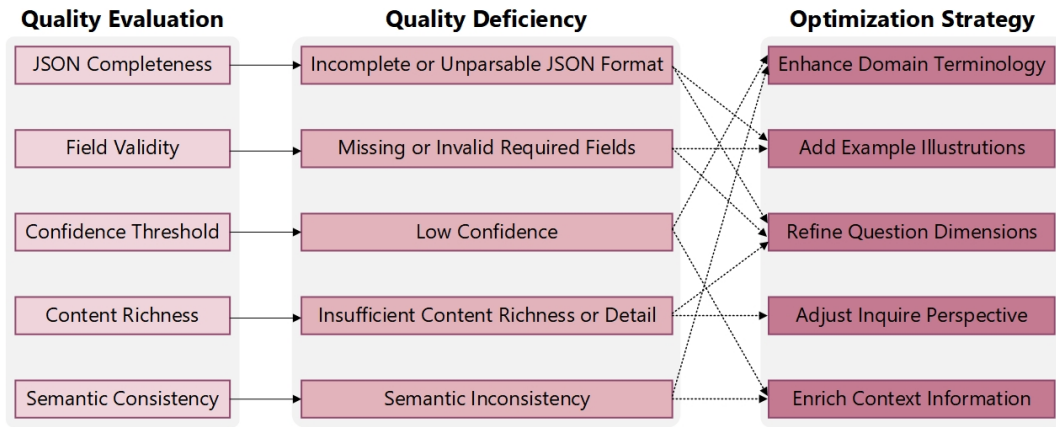


Figure 2. Discrete prompt optimization module. Mapping of quality evaluation, quality deficiencies and optimization strategies.

3.2.3. Strategy selection

The optimization strategy to be selected for specific deficiencies is shown in Figure 2. The specific implementation of the optimization strategies is as follows:

- Enhance domain terminology: Enrich the prompts with professional terms related to part attributes, mechanical assembly, and assembly processes.
- Add example illustrations: Dynamically add relevant JSON-format examples based on the type of component.
- Refine question dimensions: Break down broad questions into more specific and detailed sub-questions to improve clarity and focus.
- Adjust inquiry perspective: Reframe the question from different professional perspectives—such as design engineers, manufacturing engineers, or assembly engineers—to enhance analytical depth and accuracy.
- Enrich context information: Add information about the assembly system and technical constraints (e.g., part materials, manufacturing accuracy, assembly tools) to provide a richer reasoning context.

The SelectStrategy function implements a rule-based heuristic that, given a set of detected deficiencies and the history of previously applied strategies, selects a new strategy for prompt refinement. Specifically, the selection is made based on the current prompt, identified deficiencies, and the prior modification history. The chosen strategy is then applied to update the prompt, and the history is updated accordingly.

This iterative process forms a closed-loop feedback mechanism, where responses from the MLLM directly guide prompt refinement until either the quality criteria are met or a predefined iteration limit is reached. The matrix-guided design enables interpretable, repeatable, and targeted prompt improvements, particularly in scenarios involving weak or ambiguous instructions. The complete pseudocode is presented in Algorithm 1.

Algorithm 1 Discrete Prompt Optimization Loop

Input: Initial prompt P_0 , image I , maximum attempts N , quality threshold τ

Output: Final recognition result or failure report

```

1:  $P \leftarrow P_0, \text{history} \leftarrow [], \text{attempt} \leftarrow 0$ 
2: while  $\text{attempt} < N$  do
3:    $\text{response} \leftarrow \text{MLLM}(P, I)$ 
4:    $\text{Score} \leftarrow \text{EvaluateQuality}(\text{response})$ 
5:   if  $\text{Score} \geq \tau$  then
6:     return  $\text{Success}(\text{response}, \text{Score}, \text{attempt} + 1)$ 
7:   end if
8:    $\text{deficiencies} \leftarrow \text{Analyze}(\text{response})$ 
9:    $\text{strategy} \leftarrow \text{SelectStrategy}(\text{deficiencies}, \text{history})$ 
10:   $P \leftarrow \text{UpdatePrompt}(P, \text{strategy})$ 
11:   $\text{history} \leftarrow \text{history} \cup \{\text{strategy}\}$ 
12:   $\text{attempt} \leftarrow \text{attempt} + 1$ 
13: end while
14: return  $\text{Failure}(\text{history})$ 

```

3.3. Generation of symbolic assembly plans: planning

The planning layer is responsible for transforming the structured semantic information produced by the recognition layer into executable robotic assembly action sequences. Its core objective is to construct a symbolic assembly manual that is both physically feasible and logically coherent. This process consists of four stages: assembly constraint modeling, assembly sequence generation, step optimization, and feasibility verification. The final output is a symbolic instruction sequence organized in terms of action primitives (e.g., grasp, place, align, insert, rotate), which can be directly executed by robotic systems.

The planning layer generates a pre-placement scheme S , which encompasses all potential configurations or positions for each part within the task environment. The formula for generating the pre-placement scheme is as follows:

$$S = \text{LLM}_P(I', \text{Prompt}(I_{sp})) \quad (11)$$

where LLM_P is the pre-trained language model, and I_{sp} is the prompt for generating a pre-placement scheme.

From this scheme, and by considering the task-specific requirements and constraints, the Planner generates a sequence of operable actions, $P = \{p_1, p_2, \dots, p_n\}$, that can systematically guide the execution of the task:

$$P = LLM_P(I', D', \text{Prompt}(I_{pp})) \quad (12)$$

where I_{pp} is the prompt for generating the task plan. This process involves decomposing the high-level task into manageable subtasks, ensuring that each action within the plan is logically coherent, executable, and adaptable to evolving task conditions.

In the planning layer, the Verifier serves two primary functions. First, it verifies the correctness of the pre-placement schemes S . The Verifier checks whether the scheme meets the required criteria and rectifies any discrepancies by producing a revised scheme, denoted as S_{refined} . The verification process can be expressed as: $S_{\text{refined}} = \text{Verifier}(S)$. Upon identifying any incorrect scheme, the Verifier then incrementally updates the remaining objects in the target layout based on the generated plan. It verifies whether the planning result P satisfies predefined validation rules. If discrepancies are detected, the Verifier will re-derive the plan, generating a refined plan P' . The re-derivation of the plan can be represented by:

$$P' = \text{Verifier}(S_{\text{refined}}, P) \quad (13)$$

After successful validation, the precise plan P' will be generated, ensuring that the task plan is reliable and ready for execution.

3.4. Vision-based assembly task execution: action

For the generated assembly plans, we have selected a robot execution system based on visual positioning, whose core workflow encompasses key components such as target perception, spatial localization, task parsing, and motion control. Firstly, the system synchronously captures color images and depth images of the operating scene via an Red-Green-Blue and Depth (RGB-D) camera, providing rich visual information for subsequent three-dimensional perception. Subsequently, through image preprocessing and spatial calibration steps, the system utilizes the calibrated extrinsic parameters matrix to complete the transformation from the camera coordinate system to the robot's working coordinate system, thereby achieving precise localization of target objects in three-dimensional space.

Based on the acquired target pose information, the system receives assembly task instructions from the task planning module through the Robot Operating System 2 (ROS2) interface and parses them into executable action requirements. In terms of motion control, the trajectory planner generates an obstacle-avoiding and smooth motion path for the end-effector according to the target position and task requirements, while the trajectory controller tracks this path in real time and drives the robotic arm to complete specific assembly operations.

4. Experiments

4.1. Experimental setup

In this paper, we design two representative assembly task experiments: one focuses on the insertion and stacking of multiple types of standard structural components, and the other emphasizes the fitting

relationships among multiple heterogeneous parts, representing a coordinated multi-part assembly task. The experimental platform consists of a KUKA iiwa14 robotic manipulator, a BackYard gripper, and an Orbbec Gemini 336L camera, implemented under the ROS2 Humble framework. The MLLM used in our experiments is GPT-4o. The weight vector is defined as $\omega_i = [0.25, 0.25, 0.2, 0.15, 0.15]$, where each element corresponds to a specific evaluation dimension in Equation (1).

In Equation (8), the feature score s_f quantifies the completeness of the feature list based on the total number of provided features f , and is defined as:

$$s_f = \begin{cases} 0.4 & \text{if } f \geq 3 \\ 0.2 \cdot \frac{f}{3} & \text{if } 1 \leq f < 3 \\ 0 & \text{if } f = 0 \end{cases} \quad (14)$$

The description score s_d reflects the informativeness of the function description based on its character length d , and is defined as:

$$s_d = \begin{cases} 0.4 & \text{if } d \geq 50 \\ 0.2 + \left(\frac{d-20}{30}\right) \cdot 0.2 & \text{if } 20 \leq d < 50 \\ 0.1 & \text{if } 0 < d < 20 \\ 0 & \text{if } d = 0 \end{cases} \quad (15)$$

The function text score s_g evaluates the semantic adequacy of the function string g . It considers both the length and the presence of placeholder or template keywords:

$$s_g = \begin{cases} 0.2 & \text{if } |g| \geq 10 \text{ and 'function' not in } g \\ 0.1 & \text{if } 0 < |g| < 10 \text{ or contains template word} \\ 0 & \text{if } |g| = 0 \end{cases} \quad (16)$$

Experiment 1: the building block assembly task. This experiment aims to evaluate the system's capability in recognizing, matching, and precisely inserting regular geometric components. The assembly experiment involves approximately 531 planned assembly sequences. The robot is required to pick up 19 structural blocks of varying shapes and colors from a component area and sequentially insert them into corresponding holes on a fixed base to ultimately form a stable three-layer structure, as shown in Figure 3.

This assembly task presents a high level of complexity, requiring the robot to simultaneously satisfy multiple constraints: geometric fitting between each block and its corresponding insertion hole (e.g., circular, triangular, square), structural integrity across different assembly layers, and the sequential dependency of assembly actions. As a result, accurate task understanding is crucial for successful execution. The specific execution process is shown in Figure 4.

Experiment 2: solar panel assembly task. This experiment reflects the requirements of real-world assembly scenarios. The display diagram after assembly is shown in Figure 5. The robot captures a full-scene RGB image via an external camera and performs instance segmentation to identify each component. It then infers their structural relationships and, based on pre-calibrated eye-to-hand

extrinsic parameters, converts image pixel coordinates into the robot's working coordinate system, as shown in Figure 6.

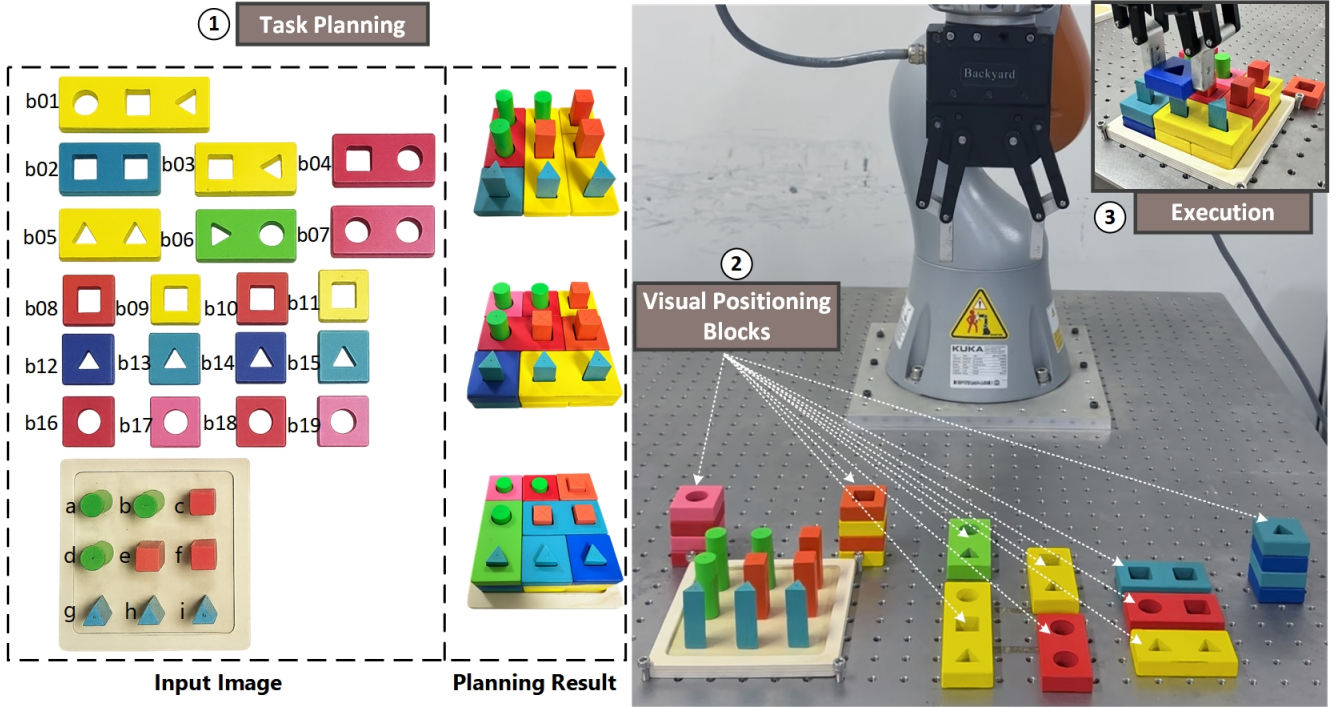


Figure 3. Experiment 1 is set up as an assembly task involving 19 blocks, each featuring holes of different shapes.

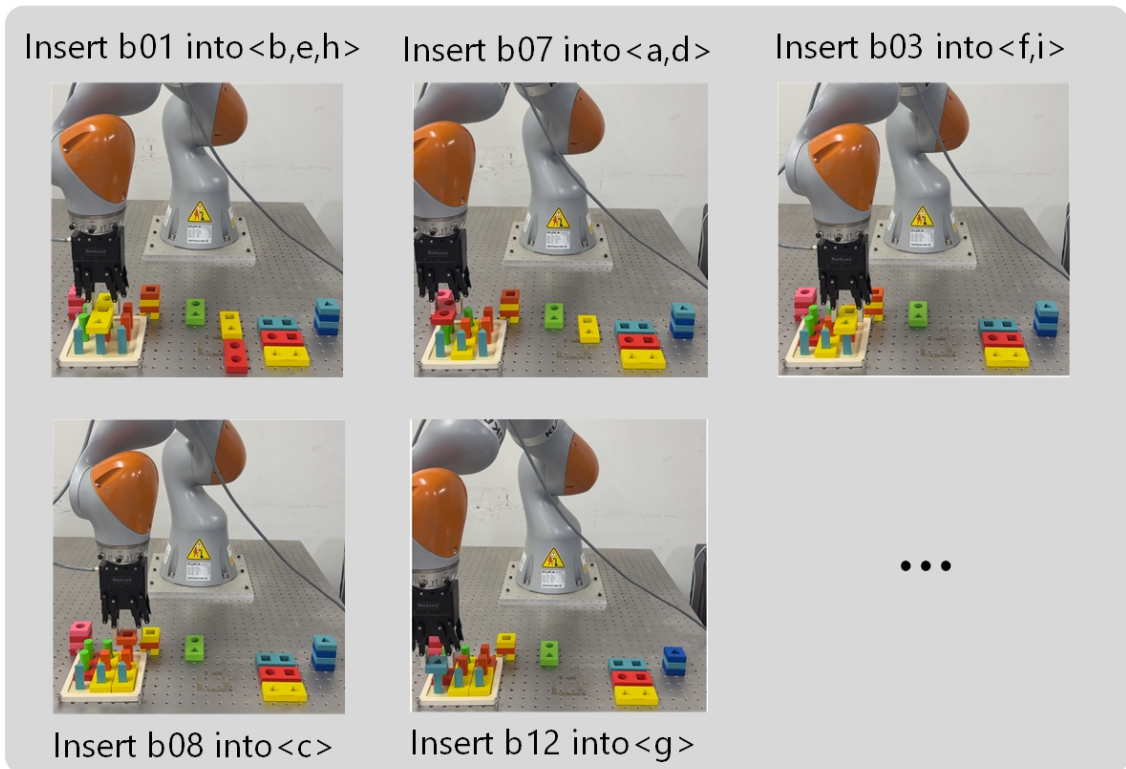


Figure 4. Experimental result diagram of the building block assembly.

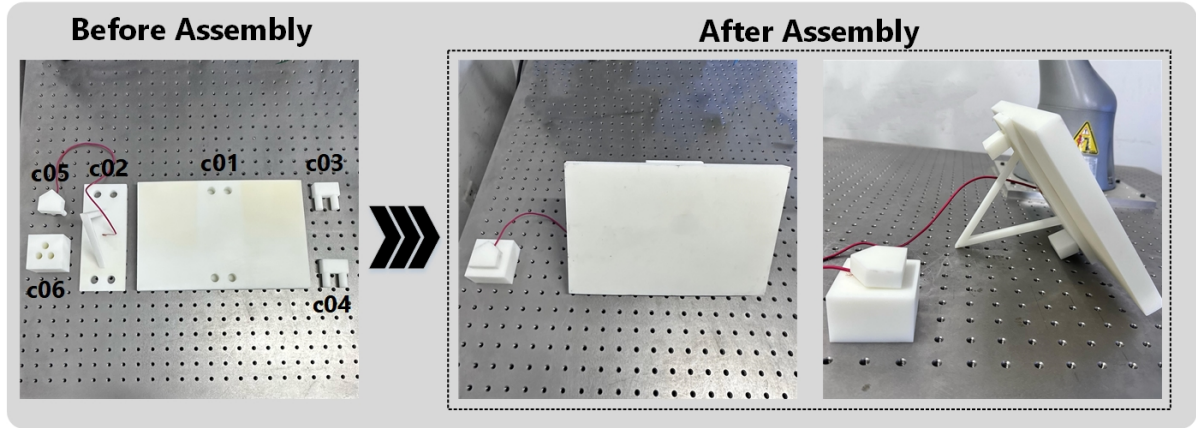


Figure 5. Solar panel assembly display diagram.

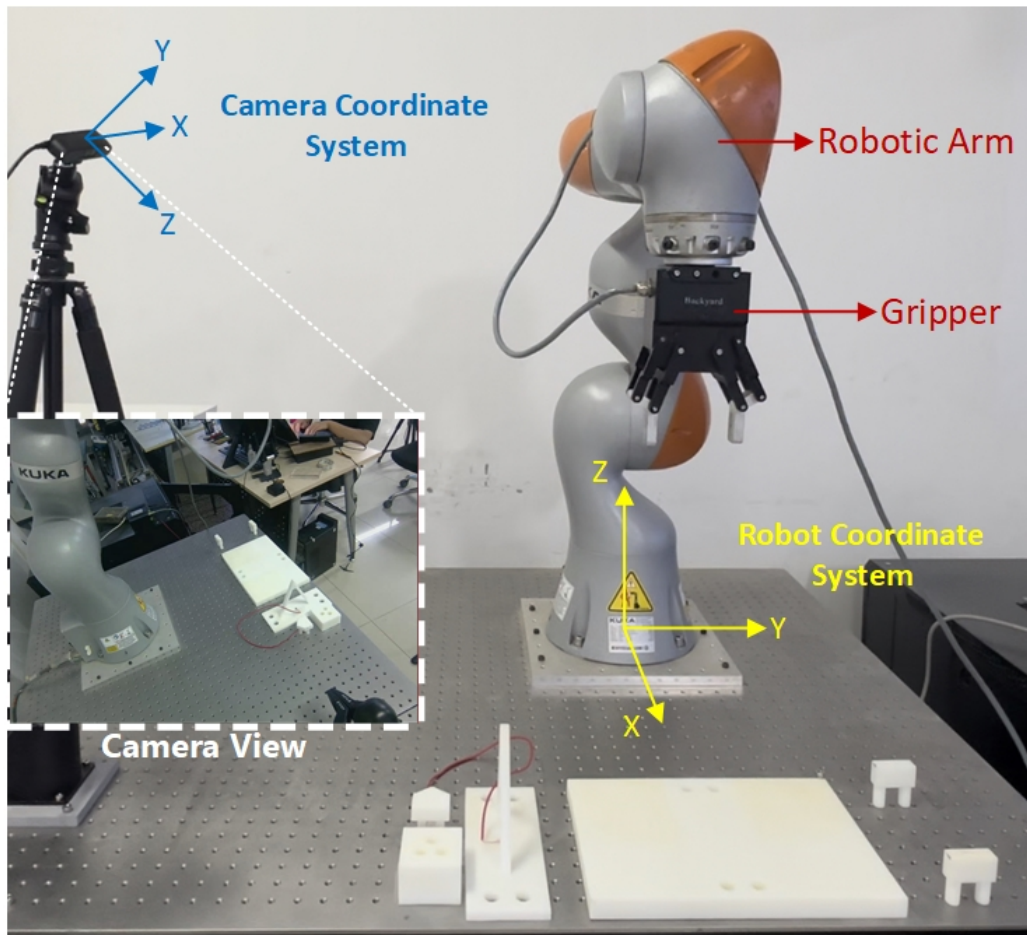


Figure 6. Experiment 2 is set up as solar panel assembly task.

Building on perception and recognition, the robot autonomously constructs an assembly topology graph of the components, infers the assembly sequence and spatial relations via a language model, and ultimately executes the composite assembly process, including insertion, nesting, and support operations. This experiment provides a comprehensive evaluation of the system's capabilities in complex structural recognition, semantic modeling, geometric registration, and fine manipulation. The specific execution process is shown in Figure 7.

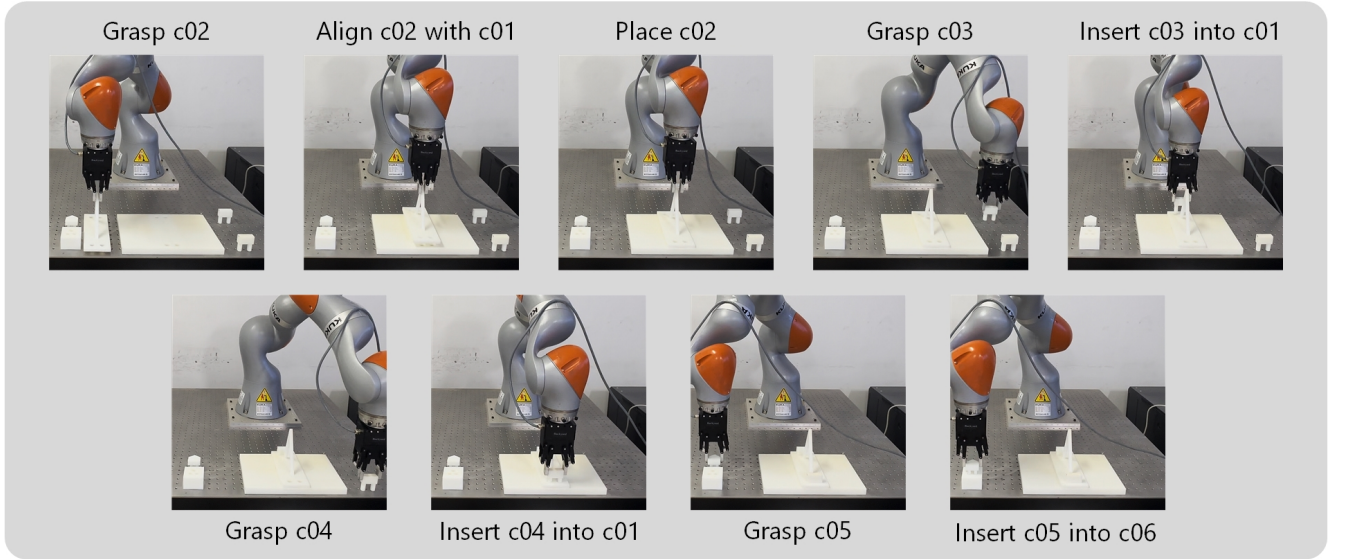


Figure 7. Experimental result diagram of multi-part collaborative assembly.

4.2. Experimental result

In the perception layer experiments, we conducted evaluations focusing on segmentation quality verification and feature extraction accuracy, to assess the system’s stability and effectiveness in instance-level component perception and structured information generation. Two key indicators were employed: mask completeness evaluation and boundary accuracy evaluation, with the IoU used as the overall metric for segmentation quality. We adopted evaluation thresholds of $\text{IoU} > 0.85$ and boundary error < 3 pixels. If the result did not meet the criteria, the system would automatically initiate a feedback loop to adaptively adjust parameters until the target precision was reached. The experimental results are shown in Table 1.

Table 1. Results of segmentation and feature extraction in the perception layer.

	Component Count (Expected/Detected)	IoU	Boundary Error (pixels)	Feature Accuracy	Iteration Count
Experiment 1	28/28	0.932	2.8	95%	4
Experiment 2	6/6	0.901	2.5	98%	1

The experimental results demonstrate that the overall output stability of the proposed discrete prompt optimization loop (Algorithm 1) increased to over 90%. Recognition stability is defined as the percentage of experimental trials in which the MLLM produces consistent, parsable, and semantically correct part recognition results (*i.e.*, correctly identifying all part types, attributes, and connection relationships) across multiple runs under identical input conditions. A trial is considered stable if the output passes all five quality evaluation criteria defined in Section 3.2.1 ($\text{Score} \geq \tau$). This metric reflects the robustness of the prompt optimization mechanism against the inherent variability of black-box MLLM responses. Table 2 presents a comparison of the system performance before and after optimization, with a total of 50 experiments conducted. In Experiment 1, a total of 216 connection relationships were identified, and seven connection relationships were recognized in Experiment 2.

Table 2. Comparison of results before and after prompt optimization.

	Experiment 1			Experiment 2		
	Before Optimization	Ours		Before Optimization	Ours	
Avg. Attempts	6.8	3.6	↓ 47%	2.3	1.2	↓ 48%
Avg. Quality Score	0.72	0.85	↑ 18%	0.72	0.89	↑ 24%
Recognition Stability	78%	95%	↑ 22%	78%	95%	↑ 22%

4.3. Result discussion

The proposed Self-Prompt SAM provides stable and accurate visual grounding across both experiments. As shown in Table 1, the system achieves IoU scores of 0.932 and 0.901, with boundary errors below 3 pixels and feature extraction accuracy above 95%, ensuring reliable inputs for subsequent processing.

The discrete prompt optimization mechanism significantly improves recognition stability and inference efficiency. As reported in Table 2, recognition stability increases from 78% to 95%, while the average number of reasoning attempts is reduced by 47% in Experiment 1 and 48% in Experiment 2.

Under geometric and task constraints, the planning module successfully generates executable symbolic assembly sequences. In Experiment 1, a valid plan is selected from 531 possible configurations, and in Experiment 2, feasible sequences involving heterogeneous parts are consistently produced.

We acknowledge that MLLMs can occasionally hallucinate non-existent parts or generate incorrect bounding boxes. In our framework, such hallucinations are mitigated through two mechanisms. First, the segmentation quality verification (IoU and boundary error checks) rejects invalid prompts—if SAM fails to produce a coherent mask from an MLLM-generated prompt, that prompt is discarded. Second, the discrete prompt optimization loop in the recognition layer further filters hallucinated parts by checking semantic consistency (e.g., whether the described part type matches extracted geometric features). In early experiments, the MLLM occasionally generated a bounding box for a “rectangular connector” in a region where only a shadow or reflection was present. SAM produced a low-IoU mask ($\text{IoU} < 0.3$), triggering the feedback loop. The system re-prompted the MLLM with a refined instruction: “Only generate prompts for parts with clear, distinct boundaries. Ignore shadows, reflections, and texture variations.”

The generated plans are executed through vision-based localization and motion control with a high overall success rate. Execution failures are rare and mainly caused by grasp instability or accumulated pose errors during insertion.

5. Conclusion

This paper addresses a central challenge in long-horizon robotic assembly: enabling robots to reliably transform visual observations into structured, executable assembly plans. While recent Multimodal Large Language Models offer strong semantic reasoning capabilities, their direct application to assembly tasks is hindered by unstructured visual inputs and unstable prompt-driven inference under black-box constraints. To overcome these limitations, we propose a PRPA framework that integrates self-prompted visual perception with discrete prompt optimization, forming a closed-loop pipeline from perception to execution.

By reformulating visual perception as a self-prompted segmentation process, the proposed approach produces stable and structured visual representations that support consistent downstream reasoning. In parallel, the discrete prompt optimization mechanism enables robust semantic understanding and relationship modeling with black-box MLLMs by iteratively refining prompts based on response quality and identified deficiencies. Together, these components allow the system to generate temporally ordered and physically feasible assembly action sequences, which are represented as symbolic assembly plans suitable for both human interpretation and robotic execution.

The effectiveness of the proposed framework is validated through two representative robotic assembly scenarios involving both homogeneous and heterogeneous components. Experimental results demonstrate improved stability in part recognition and a significant reduction in reasoning iterations, indicating that the proposed mechanisms effectively mitigate the uncertainty and variability commonly observed in MLLM-based assembly systems. These results suggest that combining structured visual grounding with quality-driven prompt refinement provides a practical pathway toward autonomous assembly planning without relying on manual prompt engineering or model fine-tuning.

Despite these advances, several challenges remain. The current framework assumes relatively static environments and does not explicitly address real-time adaptation to dynamic changes during execution. In addition, fine-grained control in contact-rich operations remains sensitive to perception noise and grasp uncertainty. Future work will explore the integration of learning-based control strategies and additional sensing modalities, such as tactile and force feedback, as well as the development of real-time replanning mechanisms to further enhance robustness in dynamic assembly settings.

Data availability statement

The data supporting the findings of this study are available from the corresponding author upon reasonable request. The experimental data include the perception, recognition, planning, and execution results obtained from the two robotic assembly tasks described in this study. No publicly archived dataset was used or generated in this work.

Declaration of Generative AI and AI-assisted Technologies

The authors declare that GPT-4o was used as an experimental component of the proposed robotic assembly framework for part recognition, semantic understanding, prompt optimization, and assembly planning. No generative AI or AI-assisted technologies were used to replace the authors' scientific judgment, data analysis, or interpretation of results. All contents of the manuscript were reviewed and approved by the authors, who take full responsibility for the final version.

Acknowledgments

This work was supported by Guangdong Major Project of Basic and Applied Basic Research under Grant 2023B0303000016, and the National Natural Science Foundation of China under Grant 92370123 and 62273280.

Authors' contribution

Qi Guo: methodology, data curation, formal analysis, software, conceptualization, investigation, writing—original draft, and writing—review and editing; Xing Liu: methodology, data curation, formal analysis, funding acquisition, supervision, conceptualization, writing—original draft, and writing—review and editing; Haitao Chang: funding acquisition, supervision, conceptualization and writing—review and editing; Zhengxiong Liu: funding acquisition, supervision, conceptualization, and writing—review and editing; Panfeng Huang: funding acquisition, supervision, conceptualization, and writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Conflicts of interest

The authors declare no conflicts of interest.

References

- [1] Lee Y, Hu ES, Lim JJ. IKEA furniture assembly environment for long-horizon complex manipulation tasks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, May 30–June 5, 2021, pp. 6343–6349.
- [2] Kroemer O, Niekum S, Konidaris G. A review of robot learning for manipulation: challenges, representations, and algorithms. *J. Mach. Learn. Res.* 2021, 22(30):1–82.
- [3] Guo Q, Wang R, Guo J, Li B, Song K, *et al.* Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv* 2023, arXiv:2309.08532.
- [4] Fernando C, Banarse D, Michalewski H, Osindero S, Rocktäschel T. Promptbreeder: self-referential self-improvement via prompt evolution. *arXiv* 2023, arXiv:2309.16797.
- [5] Deng M, Wang J, Hsieh CP, Wang Y, Guo H, *et al.* Rlprompt: optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, Abu Dhabi, UAE, December 7–11, 2022, pp. 3369–3391.
- [6] Zhang T, Wang X, Zhou D, Schuurmans D, Gonzalez JE. Tempera: test-time prompting via reinforcement learning. *arXiv* 2022, arXiv:2211.11890.
- [7] Xing Y, Chen G, Lai X, Jin S, Zhou J. Assembly sequence planning of automobile body components based on liaison graph. *Assem. Autom.* 2007, 27(2):157–164.
- [8] Kashkoush M, ElMaraghy H. Knowledge-based model for constructing master assembly sequence. *J. Manuf. Syst.* 2015, 34:43–52.
- [9] Qian J, Zhang Z, Shao C, Gong H, Liu D. Assembly sequence planning method based on knowledge and ontostep. *Procedia CIRP* 2021, 97:502–507.
- [10] Wan X, Liu K, Qiu W, Kang Z. An assembly sequence planning method based on multiple optimal solutions genetic algorithm. *Mathematics* 2024, 12(4):574.
- [11] Wu B, Lu P, Lu J, Xu J, Liu X. A hierarchical parallel multi-station assembly sequence planning method based on GA-DFLA. *Proc. Inst. Mech. Eng. C J. Mech. Eng. Sci.* 2022, 236(4):2029–2045.

- [12] Zheng Y, Chen L, Wu D, Jiang P, Bao J. Assembly sequence planning method for optimum assembly accuracy of complex products based on modified teaching–learning based optimization algorithm. *Int. J. Adv. Manuf. Technol.* 2023, 126(3):1681–1699.
- [13] Kou X, Cao Y, Qiao H. A method for intelligently optimizing hierarchical assembly structure sequences by assembly hybrid G-diagram. *Int. J. Adv. Manuf. Technol.* 2022, 122(1):27–39.
- [14] Wang Y, Geng C, Xu N. Assembly sequence optimization based on hybrid symbiotic organisms search and ant colony optimization. *Soft Comput.* 2021, 25(2):1447–1464.
- [15] Bae S, Cha H, Jiang S. Building an information modeling-based system for automatically generating the assembly sequence of precast concrete components using a genetic algorithm. *Appl. Sci.* 2024, 14(4):1358.
- [16] Ma L, Gong J, Xu H, Chen H, Zhao H, *et al.* Planning assembly sequence with graph transformer. *arXiv* 2022, arXiv:2210.05236.
- [17] Zhao M, Guo X, Zhang X, Fang Y, Ou Y. ASPW-DRL: assembly sequence planning for workpieces via a deep reinforcement learning approach. *Assem. Autom.* 2020, 40(1):65–75.
- [18] Zakka K, Zeng A, Lee J, Song S. Form2fit: learning shape priors for generalizable assembly from disassembly. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, May 31–June 15, 2020, pp. 9404–9410.
- [19] Watanabe K, Inada S. Search algorithm of the assembly sequence of products by using past learning results. *Int. J. Prod. Econ.* 2020, 226:107615.
- [20] Kitz K, Thomas U. Neural dynamic assembly sequence planning. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, Lyon, France, August 23–27, 2021, pp. 2063–2068.
- [21] Funk N, Menzenbach S, Chalvatzaki G, Peters J. Graph-based reinforcement learning meets mixed integer programs: an application to 3d robot assembly discovery. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan, October 23–27, 2022, pp. 10215–10222.
- [22] Tian Y, Willis KD, Al Omari B, Luo J, Ma P, *et al.* Asap: automated sequence planning for complex robotic assembly with physical feasibility. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan, May 13–17, 2024, pp. 4380–4386.
- [23] Dalal M, Chiruvolu T, Chaplot D, Salakhutdinov R. Plan-seq-learn: language model guided RL for solving long horizon robotics tasks. *arXiv* 2024, arXiv:2405.01534.
- [24] Zitkovich B, Yu T, Xu S, Xu P, Xiao T, *et al.* Rt-2: vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, Atlanta, USA, November 6–9, 2023, pp. 2165–2183.
- [25] Driess D, Xia F, Sajjadi MS, Lynch C, Chowdhery A, *et al.* PaLM-E: an embodied multimodal language model. In *International Conference on Machine Learning*, Honolulu, USA, July 23–29, 2023, pp. 8469–8488.