machines

MDPI

Article

GPTArm: An Autonomous Task Planning Manipulator Grasping System Based on Vision–Language Models

Jiaqi Zhang, Zinan Wang, Jiaxin Lai and Hongfei Wang *

College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao 266590, China; 202211110826@sdust.edu.cn (J.Z.); 202211110822@sdust.edu.cn (Z.W.); 202111110715@sdust.edu.cn (J.L.)

* Correspondence: skd994605@sdust.edu.cn

Abstract: The integration of vision–language models (VLMs) with robotic systems represents a transformative advancement in autonomous task planning and execution. However, traditional robotic arms relying on pre-programmed instructions exhibit limited adaptability in dynamic environments and face semantic gaps between perception and execution, hindering their ability to handle complex task demands. This paper introduces GPTArm, an environment-aware robotic arm system driven by GPT-4V, designed to overcome these challenges through hierarchical task decomposition, closed-loop error recovery, and multimodal interaction. The proposed robotic task processing framework (RTPF) integrates real-time visual perception, contextual reasoning, and autonomous strategy planning, enabling robotic arms to interpret natural language commands, decompose user-defined tasks into executable subtasks, and dynamically recover from errors. Experimental evaluations across ten manipulation tasks demonstrate GPTArm's superior performance, achieving a success rate of up to 91.4% in standardized benchmarks and robust generalization to unseen objects. Leveraging GPT-4V's reasoning and YOLOv10's precise small-object localization, the system surpasses existing methods in accuracy and adaptability. Furthermore, GPTArm supports flexible natural language interaction via voice and text, significantly enhancing user experience in human–robot collaboration.

Keywords: GPT-4V; robotic task processing framework; autonomous strategy planning; dynamic error recovery; multimodal human–robot interaction; robotic manipulation

check for updates

Academic Editor: Dan Zhang

Received: 15 February 2025
Revised: 13 March 2025
Accepted: 18 March 2025
Published: 19 March 2025

Citation: Zhang, J.; Wang, Z.; Lai, J.; Wang, H. GPTArm: An Autonomous Task Planning Manipulator Grasping System Based on Vision–Language Models. *Machines* **2025**, *13*, 247. https://doi.org/10.3390/machines13030247

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In recent years, enabling robotic arms to autonomously perform tasks based on predefined objectives has emerged as a critical research frontier, overcoming the limitations of traditional programming methods [1–4]. Researchers have explored a variety of strategies—from deep learning architectures [5–7] to human-centric interaction paradigms, such as facial recognition [8]. The rise of LLMs and VLMs has accelerated progress in this area [9,10], effectively bridging theoretical concepts with practical robotic applications. With their advanced proficiency in natural language processing and multimodal data integration, LLMs and VLMs have demonstrated transformative potential in human–robot collaboration [11,12] and cognitive task reasoning [13,14].

For robotic manipulation systems, accurately executing natural language instructions requires aligning linguistic specifications with kinematic constraints. Early studies [15,16] focused on syntactic alignment between language inputs and predefined action templates, while subsequent work [17,18] integrated AI models for spatial reasoning and command-to-action translation. Implementations such as that in [19] demonstrated a direct mapping

from linguistic instructions to robotic trajectories. However, these systems suffered from weak error-handling strategies, relying heavily on command filtering and hard-coded motion primitives, which limited their operational robustness and practical applicability. Modern LLMs address these shortcomings through contextual language understanding and compositional task reasoning, establishing themselves as fundamental components for next-generation robotic intelligence [20]. The fusion of LLMs [21] with computer vision systems [22] has significantly advanced robotic competencies in complex, environment-aware planning [23], facilitating dynamic strategy generation via multimodal context interpretation [24–26] and high-precision action sequence formulation [27,28].

Recent advancements in robotic task processing frameworks have significantly enhanced autonomous systems by leveraging VLMs and LLMs. For example, RT-2 [29] showcases the ability to transfer web-derived knowledge to robotic control, enabling more sophisticated task execution, while interactive task planning [30] and ontological frameworks [31,32] enhance adaptability in dynamic, unstructured environments. These developments have improved robotic reasoning and decision-making, yet VLM-driven systems continue to encounter significant deployment challenges. Semantic disconnects between visual perception outputs and task-specific contexts, coupled with spatial localization inaccuracies, frequently result in flawed task planning and execution failures. Such issues often stem from the uncritical reliance on AI-generated hypotheses without sufficient environmental validation. Current solutions, such as hybrid architectures integrating VLMs [33–35] with verification modules [36–38] and 3D perception enhancements via stereo vision systems [39], have improved grasping precision and task accuracy, though they introduce high hardware costs and computational complexity. Approaches employing fixed monocular cameras [40] provide a cost-effective solution for environmental awareness [41] but suffer from accuracy degradation due to viewpoint dependency and limitations in adaptive planning for dynamic settings, making scalable VLM integration in robotics difficult. Recent frameworks [31,42] have advanced task processing architectures, yet persistent challenges in spatial reasoning and error recovery highlight the need for more robust approaches. Prior work [43] demonstrates motion plan refinement based on human–robot collaboration (HRC); however, it requires extensive training data and lacks cross-domain generalization. Table 1 provides a comparative analysis of current robotic task processing frameworks, detailing their strengths and limitations.

Table 1. The advantages and disadvantages of current robotic task processing frameworks.

Reference (Year)	Advantage	Disadvantage
[14]-(2024)	Integrates segmentation and grasp synthesis to eliminate action model reliance.	Segmentation/synthesis errors may cascade.
[28]-(2024)	Proposes a Robot-GPT interaction framework with code generation and error correction.	Visual feedback absence restricts error comprehension.
[31]-(2024)	Combines internal/external error correction mechanisms.	Only grasps trained objects.
[43]-(2024)	Hierarchical planning decomposes long-term tasks.	Ineffective control of errors arising during execution.
[44]-(2024)	Conditional action sampling generates VLM-based targeted actions.	Predicted video–action mismatch degrades evaluation.

To overcome the limitations in prior work—including extensive training data, poor cross-domain generalization, inadequate spatial reasoning, and error recovery—we present GPTArm, a VLM-driven robotic system that highlights autonomous task planning capabilities through our robotic task processing framework (RTPF). The RTPF introduces three key mechanisms: (1) autonomous task planning, which dynamically generates execution strategies based on user-defined objectives, enabling real-time adaptation to task and

environmental changes; (2) dynamic error recovery, which self-corrects plans through closed-loop feedback in dynamic environments; and (3) multimodal context integration, which integrates vision, language, and spatial data to achieve context-aware precision. By directly addressing VLMs' limitations in 3D spatial cognition and compositional reasoning, the RTPF achieves a 91.4% success rate in standardized benchmarks, thereby demonstrating its capability for autonomous task execution in real-world scenarios.

The key contributions of this work are as follows:

1. Development of the RTPF: We propose a novel framework integrating GPT-4V with robotic arm control systems, leveraging environmental data from vision sensors and advanced reasoning capabilities. The RTPF implements a hierarchical decomposition paradigm that accomplishes the following:
   - Autonomously plans task execution based on user-defined objectives, generating context-aware subtasks in real time.
   - Selects optimal motion primitives from a predefined library to ensure precise and efficient operations.
   - Validates and corrects execution outcomes through real-time closed-loop feedback, ensuring robust performance in dynamic environments.
2. Environment-Aware Adaptive Planning: We introduce a hybrid natural language processing pipeline that autonomously extracts semantic objectives from unstructured instructions and generates environment-adaptive strategies, supported by multimodal voice–text interaction.
3. Cross-Model Performance Benchmarking: A comprehensive evaluation of state-of-the-art VLMs (Fuyu-8B, Qwen-vl-plus, GPT-4V) reveals that GPT-4V excels in task decomposition and execution accuracy under dynamic conditions. Experimental results highlight the framework's exceptional versatility, achieving consistent performance across diverse tasks—from simple object grasping to complex, multi-step assembly operations.

2. Materials and Methods

GPTArm is an AI-integrated, versatile robotic arm system designed to overcome the limitations of conventional robotic arms. Although a wide variety of robotic arms are available on the market—each differing in type, model, and functionality—the development processes and design complexities vary significantly, thereby limiting their universality and interoperability. The primary objective of GPTArm is to address these challenges by enabling heterogeneous robotic arms to execute identical tasks through a unified task processing framework.

2.1. RTPF: A Hierarchical Framework for Environment-Aware Task Planning

The core of GPTArm is the RTPF, a novel hierarchical framework for autonomous task execution. Integrated with robotic arm control interfaces, it forms a closed-loop workflow through four specialized computational layers. By assigning complex task planning and multimodal data processing to GPT-4V and perception/execution to robotic hardware, the RTPF creates a human–robot intelligence collaboration paradigm.

The RTPF is primarily inspired by the hierarchical structure from [43] and the error correction mechanism from [31]. Through architectural improvements, it effectively addresses limitations such as the lack of dynamic error recovery and dependency on pre-trained datasets. Specifically designed as a cross-platform task processing framework, it aims to enhance universality across diverse robotic arm systems while maintaining hardware-specific motion capabilities. Figure 1 presents the detailed structure and operational workflow of the RTPF.

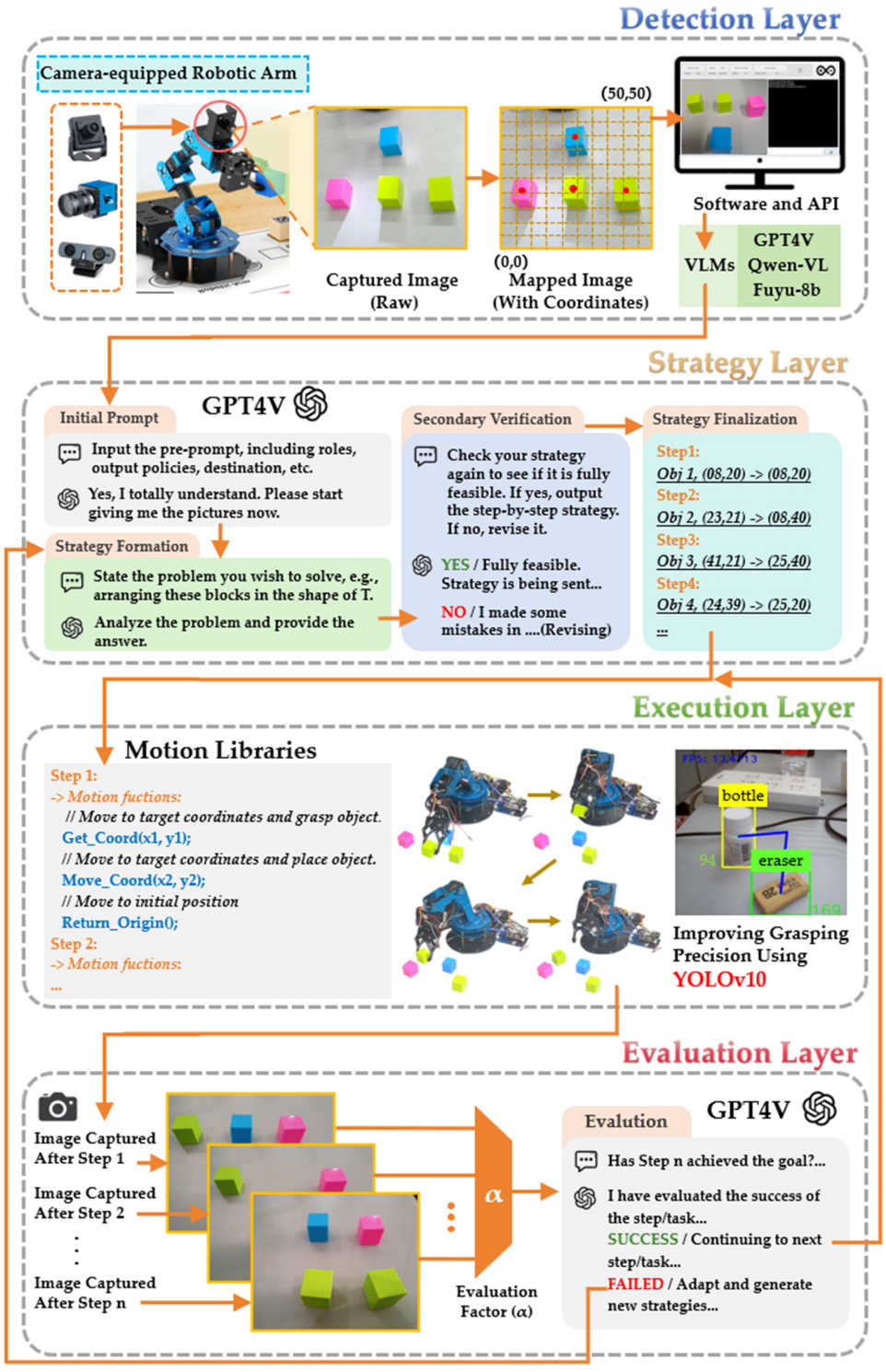

Figure 1. Detailed architecture of the RTPF, illustrating the hierarchical layers for autonomous task planning and execution.

The multi-layer architecture of the RTPF operates through coordinated interactions among the following:

1. Detection Layer: Implements real-time sensor fusion and environmental mapping.
2. Strategy Layer: Employs GPT-4V for contextual task decomposition and constraint satisfaction.
3. Execution Layer: Converts abstract plans into hardware-specific control signals.
4. Evaluation Layer: Provides continuous performance monitoring and error compensation.

2.1.1. Detection Layer

The detection layer, consisting of a vision module and a processing unit, captures visual data from the robotic arm's operating environment and transmits it to GPT-4V for subsequent processing and analysis. This ensures accurate environmental detection to support downstream task planning and execution.

In our system, the vision module uses a USB Video Class (UVC)-compliant camera mounted beneath the robotic arm's gripper, moving with the arm, while fixed overhead cameras provide stable, undistorted views for simple grasp localization—they are less effective in scenarios involving small objects or constrained workspaces. In contrast, an integrated camera dynamically tracks target objects and adjusts its viewpoint in real time, making the system adaptable to a wide range of operational demands.

However, mounting the camera on the robotic arm introduces challenges. Tilted viewpoints can complicate object localization, especially with partial object occlusion. Additionally, arm-movement-induced viewpoint shifts may result in positioning inaccuracies. These issues are mitigated in the strategy layer, where GPT-4V intelligently corrects and optimizes the initial data, substantially enhancing grasping accuracy. Once the camera captures a raw image x^t of the environment at time t, the system applies a series of preprocessing operations—collectively denoted by $f_{process}$—to yield a processed image x_p^t:

$$x_p^t = f_{process}(x^t) \tag{1}$$

These operations include basic image enhancement, noise reduction, and partitioning the image into a 50×50 grid to discretize the coordinate space for subsequent analysis. The processed data are then forwarded to GPT-4V for recognition and interpretation, thereby informing both the task planning in the strategy layer and the motion control in the execution layer. By organizing environmental sensing in this layered manner, the detection layer establishes a robust data foundation for subsequent task execution.

2.1.2. Strategy Layer

The strategy layer, built upon GPT-4V, serves as the core mechanism for task analysis and planning. Instead of just processing fixed instructions, it interprets user intentions, considers context, and designs efficient steps. Through well-configured prompts and formats, GPT-4V adapts flexibly to a wide variety of tasks and interaction requirements.

In the presented framework, GPT-4V typically receives two types of inputs. (1) Role definition specifies domain expertise and decision-making style (e.g., acting as an experienced designer for block arrangement tasks to evaluate block positions). (2) The standardized response format ensures consistent data exchange with the robotic arm via coordinate–action data. GPT-4V segments images into coordinate systems and returns precise positional data for translating strategies into control operations.

(1) Task Decomposition.

Using processed visual data and high-level task goals, GPT-4V decomposes each task into manageable subtasks. For instance, in a block arrangement task, it identifies current block positions, determines target coordinates, and generates grasp–move–place actions to achieve the desired configuration. This decomposition converts abstract objectives into a structured set of subtasks.

(2) Strategy Planning.

Once the task is decomposed, GPT-4V formulates a strategy represented as a sequence of action steps. Let I denote the set of additional constraints or parameters (such as workspace boundaries or object properties). GPT-4V then generates a plan p:

$$p = GPT - 4V\left(x_p^t,\ I\right) = \{s_1,\ s_2,\ \ldots,\ s_n\} \tag{2}$$

Each step s_i specifies both the coordinates and the required action (for example, "grasp the object at (x_i, y_i)" or "place the object at (x_j, y_j)"). By comparing the current configuration of objects with the desired outcome, GPT-4V produces an ordered list of grasping and placing commands. If a target position is occupied, the plan adjusts to prevent interference, ensuring a coherent workflow with minimal unnecessary maneuvers. Finalized action sequences are translated into hardware-compatible control protocols, enabling accurate execution and enhancing task robustness and efficiency.

(3) Secondary Verification.

Strategies generated by GPT-4V are not always correct. Incorrect ones may harm objects or even damage robots [31]. To address potential inaccuracies in object localization or suboptimal sequencing, the system incorporates a secondary verification. We define a function $F(p)$ that maps an initial strategy p to a final, validated strategy p_{final} as follows:

$$p_{final} = \begin{cases} p & \text{if } validate_strategy(p) = \text{True} \\ error_correct(p) & \text{otherwise} \end{cases} \tag{3}$$

In this formulation, if the initial strategy p meets all feasibility criteria, it is retained; otherwise, the function *error_correct*(p) procedure adjusts and refines the plan to accommodate constraints. Experimental results indicate that this secondary verification stage significantly improves task success rates while maintaining practical response times (detailed results are presented in Section 3.2.1). Although additional iterations could further reduce residual errors, our experiments demonstrate that more than two verification loops yield only marginal improvements relative to the increased processing time.

By integrating task decomposition, coordinated planning, and a secondary verification mechanism, the strategy layer facilitates effective task execution. GPT-4V's constraint-aware reasoning, combined with these design choices, enhances success rates and reduces end-user programming effort.

2.1.3. Execution Layer

The execution layer transforms the optimized instructions from the strategy layer into precise robotic arm movements. In the GPTArm system, this layer achieves accurate control by integrating a corrected motion library with robust control functions.

The motion library functions as a comprehensive database containing predefined motion functions, path planning algorithms, and execution routines tailored to various robotic arm models. It provides motion paths accounting for differences in motion control and hardware structure across different systems. These motion functions include inverse kinematics (IK) solvers [45], path planning modules, and interference correction routines that facilitate the conversion between joint space and Cartesian space [46], ensuring smooth and accurate execution of grasping and placement operations.

After task decomposition, GPT-4V forwards the segmented strategies to the computer. The computer then parses the current and desired object coordinates and transmits these parameters to the robotic arm control system. Based on this information, the robotic arm selects appropriate motion functions from the motion library, formally represented as:

$$code = extract_motion_code(p) \tag{4}$$

where *extract_motion_code*(p) is a function that maps the strategy p to the corresponding motion code. This motion code specifies the path planning algorithm and control parameters

to be executed, thereby ensuring that the robotic arm performs the intended motion with high precision.

The execution process operates as a closed-loop control system, with continuous feedback from each step relayed to the evaluation layer for real-time verification of task completion and accuracy. This design ensures motion stability and precision while enabling rapid deviation correction.

GPT-4V's versatility eliminates the need for object feature pre-training in conventional grasping tasks. However, predefined motion routines lack precision for low-profile items like small nuts, constrained by the robotic arm's mechanical limits and the motion library's resolution. To address this issue, we incorporated YOLOv10 for dynamic calibration. YOLOv10 demonstrates strengths in speed, accuracy, adaptability, and small-object detection, and it is widely used in object recognition for robotic arm grasping [47]. Compared to prior versions like YOLOv8, it brings a 2.6% improvement in the F1 score while cutting parameters by 37% [48]. When the robotic arm nears the target area, YOLOv10 is activated to detect the object in real time and provide precise center coordinates. Based on this visual feedback, the robotic arm continuously adjusts its posture and position, gradually aligning the end effector with the object's center. This ensures the grasping operation is performed at the optimal angle and position, significantly improving the grasp success rate. The online correction mechanism, combining visual feedback and real-time adjustments, overcomes the limitations of traditional fixed-coordinate localization methods, enabling high-precision operation even in complex, dynamic environments. Detailed information is provided in Appendix A.

Overall, the design of the execution layer emphasizes the seamless integration of robotic arm control with intelligent planning from the strategy layer. This integration enables successful routine grasping and placement tasks and flexible adaptation to dynamic changes like target object displacement. By coupling efficient motion planning algorithms with diverse feedback control mechanisms, the execution layer achieves robust and efficient task execution across a range of operating conditions.

2.1.4. Evaluation Layer

Due to large-scale AI models' reasoning and decision-making limitations, the instructions output by the strategy layer may contain errors affecting robotic arm operations. To mitigate this risk, we introduced an evaluation layer that performs real-time validation of the results at each execution step, reducing failure risks from erroneous commands.

The evaluation layer captures and assesses the instruction outcomes generated by the strategy layer after each task step. After robotic arm task completion, the vision module captures operational state images for GPT-4V analysis. By comparing the actual outcome with the expected task goals, GPT-4V evaluates the accuracy and completeness of the executed step. If the evaluation confirms that the task has met the expected criteria, the system proceeds to the next step. Otherwise, GPT-4V analyzes the error, regenerates the corresponding operation steps, and sends the adjusted instructions back to the execution layer for correction. This closed-loop feedback mechanism significantly enhances both the grasp success rate and the overall execution accuracy of the robotic arm.

Similar evaluation setups have been reported in recent research [31]. However, in multi-step workflows, errors occurring in earlier steps may lead to cumbersome error correction processes if relying solely on final outcome assessments. Additionally, this approach fails to provide an effective measure of the system's performance during execution. Therefore, we introduced an "evaluation factor" to enable flexible adjustment of the evaluation frequency and stringency. Denoted by α (with $0 < \alpha \leq 1$), this factor regulates evaluation frequency across operational steps. Higher α values trigger rigorous per-step

evaluations for precision, while lower α values limit evaluations to post-task assessments to enhance efficiency. This parameter strikes a balance between task precision and overall efficiency, adapting to various operational needs.

To formalize this mechanism, let us define the following:

- N denotes the total number of task steps;
- α represents the evaluation factor;
- m is the number of steps designated for correction.

The number of steps requiring evaluation is computed as:

$$m = \lceil N \times \alpha \rceil \tag{5}$$

where $\lceil . \rceil$ denotes the ceiling function, ensuring that mmm is an integer. The criterion for determining whether the current step n requires evaluation is defined by the function:

$$E(n,\ N,\ \alpha) = \begin{cases} \text{True} & \text{if } n = N \text{ or } n = \left\lceil \frac{n \times N}{m} \right\rceil \\ \text{False} & \text{otherwise} \end{cases} \tag{6}$$

In this formulation, the final step ($n = N$) is always evaluated, and the evaluation steps are uniformly distributed according to the computed m.

The evaluation layer introduces an adaptive correction mechanism, ensuring high operational precision and reliability even in complex dynamic environments. By combining visual feedback with intelligent analysis, it enhances task success rates and mitigates strategy-layer errors, improving system robustness and adaptability. The RTPF's layered approach decouples perception, planning, and execution, forming a robust closed-loop system.

Pseudocode (see Algorithm 1) is provided to illustrate the overall process, where x^t is the raw image captured by the camera at time t.

Algorithm 1: RTPF

Initialize camera C, task M, natural language instruction I, evaluation factor α
while task not completed **do**
  Capture raw image $x^t = C(t)$
  Preprocess image $x_p^t = f_{process}(x^t)$ // Noise reduction, grid partitioning
  $p = GPT\text{-}4V(x_t, I)$ // Generate initial plan
  if *validate_strategy*(p) == False **then**
    $p_{\text{final}} = error_correct(p)$ // secondary verification
  else
    $p_{\text{final}} = p$
  end if
  $code = extract_motion_code(p, M)$ //Map to motion functions
  Execute *code*, update gripper pose
  $x_{new} = C(t_{new})$
  for k = 1 to N **do** //N: total steps in p_{final}
    if $E(k, N, \alpha)$ = True **then**
      $result = GPT\text{-}4V(x_{new}, p_{\text{final}})$ //Apply evaluation feedback
      if *result* = False **then**
        Trigger feedback loop: send x_{new} to strategy layer for re-planning
      end if
    end if
  end for
end while
return Success

2.1.5. Pre-Instructions of RTPF Layers

Figure 2 illustrates the mapping between the predefined requirements and the pre-instructions of each RTPF layer.

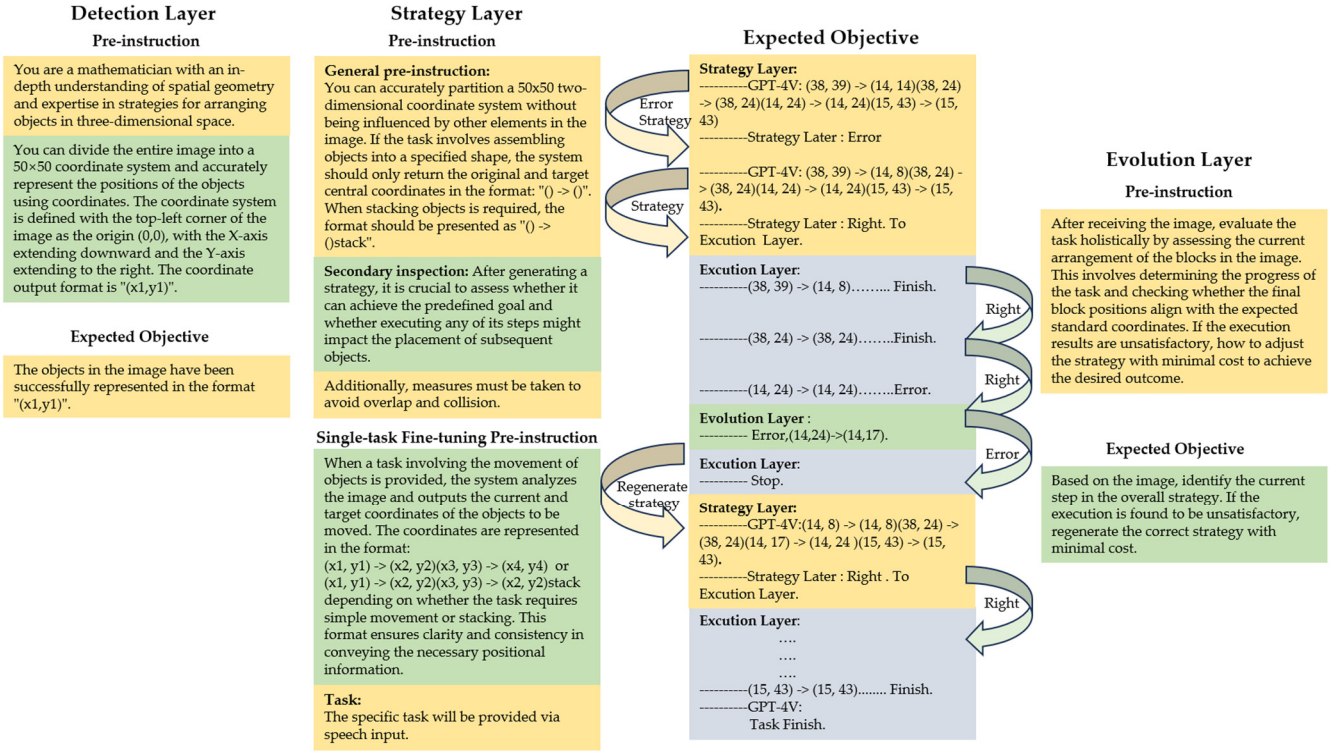

Figure 2. Predefined requirements and functional mapping across RTPF layers.

In the detection layer, the role-defining and task-specification pre-instructions are crucial. They assign GPT-4V the role of a mathematician well-versed in spatial geometry, enabling precise image coordinate processing by partitioning the image into a 50×50 coordinate system and specifying object position output formats. The ultimate objective is to empower GPT-4V to accurately represent object positions and occupied coordinate ranges.

The strategy layer encompasses three types of pre-instructions: general, secondary, and task-specific fine-tuning pre-instructions. The general pre-instructions guide overall coordinate system partitioning and object-arrangement formats. The secondary pre-instructions evaluate strategy feasibility and prevent overlaps and collisions. The task-specific fine-tuning pre-instructions are customizable for specific tasks, like prompting GPT-4V to move blocks with coordinate conflicts to unoccupied areas in a building block placement task, aiming to generate accurate and viable task-execution strategies.

The execution layer acts as a bridge, receiving instructions from the strategy layer and translating them into physical movements of the robotic arm. By integrating a motion library and control functions, it converts optimized instructions into smooth and accurate grasping and placement operations, ensuring compliance with the strategy layer's directives.

Lastly, the evolution layer is equipped with holistic evaluation and adjustment-oriented pre-instructions. These pre-instructions direct GPT-4V to assess the overall task based on the visual input from the image, monitor the progress, and determine whether the final positions of the objects match the expected standards. In the case of any discrepancies, GPT-4V adjusts the strategy at minimal cost to rectify task-execution deviations and achieve the desired outcome.

The hierarchical design of the RTPF offers two critical advantages. First, it enables automatic strategy planning by allowing GPT-4V to decompose complex tasks into executable sub-steps using real-time environmental inputs. Second, it incorporates a verification process supported by closed-loop feedback that detects and corrects errors in the initial plan. Together, these mechanisms enhance the system's adaptability, precision, and scalability for diverse robotic manipulation tasks.

The RTPF employs a modular architecture to ensure compatibility with diverse robotic systems rather than being restricted to our specific robotic arm. It can be seamlessly integrated into any task processing system meeting three core requirements: (1) a camera (any type) for real-time visual feedback; (2) a motion library enabling precise navigation to coordinates within the camera's field of view; and (3) network connectivity (low-cost implementations use devices like ESP32 microcontrollers instead of computers, reducing system costs by up to 90%).

While supporting robotic arms across cost ranges, the coordinate-based operation limits deployment to continuous planar workspaces directly in front of the robotic arm. This constraint arises from the system's reliance on 2D Cartesian coordinate mapping for motion planning and object localization.

2.2. Intelligent Human–Robot Interaction via Integrated Voice and Text Command

In the context of human–robot interaction, we integrated LLMs with advanced visual processing technologies to significantly enhance the flexibility and accuracy of robotic arms during task execution. Robotic arms equipped with the RTPF demonstrate clear advantages in human–robot interaction and target task completion compared to traditional systems without AI integration (see Figure 3 for an overview of the architecture).

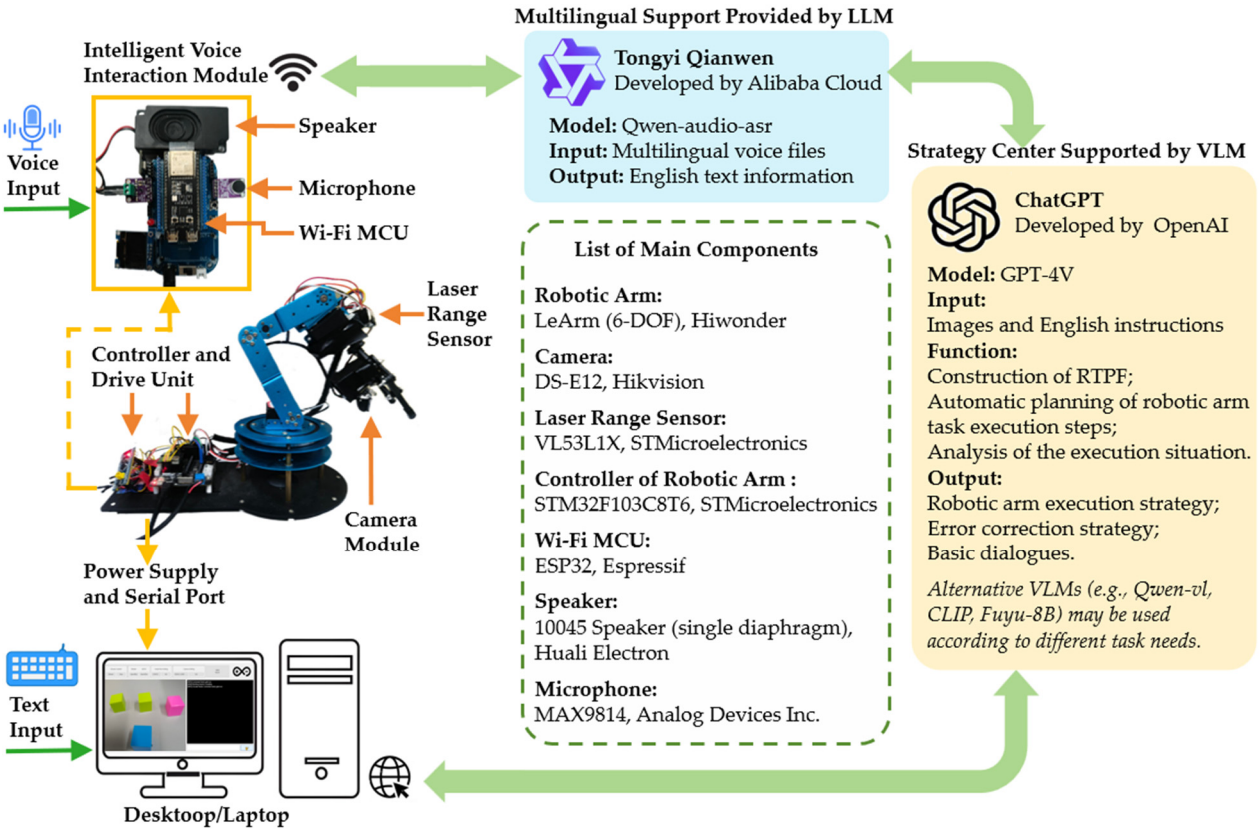

Figure 3. Overall architecture of the GPTArm integrating a VLM (GPT-4V) and providing multimodal voice–text interaction.

Traditional robotic arms face two primary challenges in instruction processing and object recognition:

1. Strict Dependence on Precision in Instructions: Conventional systems can only respond to explicit, precise operational commands and are incapable of handling ambiguous or vague natural language expressions.
2. Limited Recognition Capability: Traditional robotic arms rely on pre-trained image recognition models and can only recognize objects included in their training datasets, rendering them ineffective at identifying new or unseen objects.

To overcome these limitations, we integrated the Qwen-audio-asr model from the Tongyi Qianwen platform to develop a comprehensive speech input–output system supporting multi-language speech input. Research demonstrates that Qwen-Audio surpasses models like SALMONN in speech recognition. Integrating with LLMs [49], it supports multiple tasks, including automatic speech recognition (ASR), speech translation, and audio question answering [50]. Speech commands in various languages are first converted into English text and then processed by the GPT model. When faced with ambiguous instructions, GPT synthesizes contextual information to generate specific execution decisions. For example, in response to a vague command such as "I feel a bit unwell", GPT can infer appropriate task steps based on the surrounding context. After processing, GPT's textual output is converted back into speech for real-time voice feedback, ensuring a seamless bidirectional communication channel.

The voice output module supports real-time human–robot communication by providing natural language feedback on execution states. For instance, upon completing a grasping task, the system announces "Task completed", while failure to detect a target object prompts the message "No target object detected within the field of view". It also offers customizable responses and simple daily conversations, enhancing the bidirectional user experience.

A notable feature of our design is zero-shot object grasping. Due to the inherent recognition capabilities of GPT-4V, the system reliably identifies and locates novel objects by analyzing the coordinate positions of objects captured in images. Consequently, it generates a corresponding grasping strategy that is transmitted to the robotic arm's control unit, which then selects an appropriate motion function from the motion library to execute the grasp. This approach enables the robust performance of the system to handle un-trained objects.

By combining the natural language processing capabilities of large language models with high-precision object detection and multimodal voice–text interaction, our system not only interprets and executes ambiguous commands but also improves task accuracy through real-time visual and auditory feedback. This integrated approach significantly enhances the robotic arm's adaptability and flexibility in complex environments, laying a robust foundation for future advancements in human–robot collaboration.

3. Results

3.1. Experiment Setup and Evaluation

3.1.1. System Setup

All experiments in this study were conducted using a six-degrees-of-freedom (DoF) LeArm robotic arm (one rotating base, four movable joints, and a gripper). This integrated robotic platform, combined with various VLMs, is termed GPTArm. Our primary VLM is GPT-4V (provided by OpenAI), with Qwen-vl-plus (from the Tongyi Qianwen platform) and Fuyu-8B (accessed through the Baidu Smart Cloud) used for supplementary comparisons. Each VLM is incorporated via dedicated API endpoints, enabling GPTArm to process visual or textual inputs and generate robotic commands.

A standard 2D camera mounted beneath the gripper captures real-time workspace images. These visual data are processed by the RTPF to segment scenes and map them to a grid-based coordinate system as needed. The experiment area itself is a flat work surface where both target objects and distractor items are placed in varied arrangements. Trials start with the arm in a neutral pose and end upon task completion or failure under constraints.

3.1.2. Evaluation Metrics

To assess the system's effectiveness under various conditions and difficulty levels, we adopt the following metrics [27].

- Success Rate (SR).
  The proportion of tasks in which all intended goal conditions are met by the end of the robotic arm's operation. For tasks comprising multiple sub-steps, a single failure in any crucial step renders the entire task unsuccessful.
- Executability (EXEC).
  A measure of how many planned action steps can be physically executed without error. Formally,
  $$EXEC = \frac{N_{correct}}{N_{total}} \tag{7}$$
  where $N_{correct}$ represents the number of correctly executed steps and N_{total} means the total steps in the plan.
- Goal Condition Recall (GCR).
  An indicator of how many final goal conditions are satisfied relative to the total conditions specified. *GCR* is computed as follows:
  $$GCR = 1 - \frac{UGC}{TGC} \tag{8}$$
  where *TGC* is the total number of goal conditions (e.g., precise object positions/orientations) and *UGC* is the number of these conditions still unmet after execution. A *GCR* value of 1 implies that all intended goal states were realized, whereas lower values reflect partial fulfillment or missed targets.

3.2. RTPF Validation: Task-Specific Performance Analysis

To evaluate GPTArm's performance across a range of manipulation challenges, we designed ten distinct tasks, each focusing on grasping and arranging common objects (e.g., medicine bottles, nuts, erasers, blocks). These tasks were categorized into four difficulty levels, reflecting both the complexity of the actions involved and the precision demands of sequential operations.

- B (basic) tasks entail direct single-object grasps under relatively clear conditions.
- I (intermediate) tasks incorporate either two-step sequential operations or target objects located in less favorable viewing conditions.
- A (advanced) tasks involve multi-step planning or partial re-planning upon receiving updated instructions
- HA (highly advanced) tasks increase the spatial or sequential complexity further, often requiring stacking operations and dynamic corrections.

Table 2 provides an overview of the tasks, their difficulty classification, and a brief description of key requirements. For the video materials related to the experiment, please refer to Supplementary Video S1.

Table 2. Task classification and complexity levels.

No	Task	Difficulty	Description
Task 1	Medicine Bottle Grasping	B	Pick up a single bottle from the workspace.
Task 2	Nut Grasping		Pick up a small nut from the workspace.
Task 3	Block Grasping		Pick up a block from the workspace.
Task 4	Circular Grasping	I	Circularly grasp all the small nuts from the workspace, where there may be slight occlusion or a complex environment.
Task 5	Edge-of-View Object Grasping		Requires careful localization when objects are only partially visible.
Task 6	T-Shaped Block Arrangement	A	Requires placing multiple blocks in a specific geometric configuration; demands accurate positioning and orientation handling.
Task 7	Command-Driven Regrasping Following Misinterpretation		Simulates an instruction correction: pick up one object (e.g., a bottle) then replace it upon revised instructions (e.g., "move the charger instead"). Tests dynamic re-planning.
Task 8	Partial T-Shaped Rearrangement		Involves moving or removing some blocks from a T-configuration and re-stacking them according to new instructions.
Task 9	T-Shaped Construction with Additional Block Stacking	HA	Combines arrangement (forming the T) with vertical stacking of new blocks. Requires stable, multi-step sequencing with minimal positional error.
Task 10	Multiple Block Stacking		Requires sequentially stacking three or more blocks with distinct target positions, potentially re-planning if partial occlusions or collisions occur.

We evaluated the RTPF on four representative tasks of increasing difficulty: (B) task 1, medicine bottle grasping, (I) task 4, circular grasping, (A) task 6, T-shaped block arrangement, and (HA) task 10, multiple block stacking. Each time GPT-4V produces a new strategy, it encapsulates the object's current and intended coordinates in a standardized transmission format, which is then relayed to the experimental apparatus. This approach ensures that the execution layer can systematically drive the robotic arm and enables human operators to reliably assess the plan's correctness.

3.2.1. Strategy Layer Performance Evaluation

At the beginning of each trial, GPT-4V receives the current camera image and a concise task description (e.g., "Grasp the Medicine Bottle" or "Arrange Blocks to Form a T-Shape") and then produces a textual strategy of the required coordinate mappings. To address potential self-detection limitations, the framework automatically performs a secondary verification in which GPT-4V re-examines the proposed strategy. If inconsistencies (e.g., conflicting coordinates or missing steps) are detected, a revised strategy is output; otherwise, the original plan is validated. Human reviewers monitor these strategies to gather accurate statistics but do not intervene in execution decisions.

To gauge the reliability of this strategy generation and revision process across tasks of varying complexity, we performed 210 trials per task (B, I, A, HA) 210 times. For each trial, we recorded whether GPT-4V's secondary verification triggered revision and whether those revisions resolved the original inconsistencies. This procedure yielded a comprehensive dataset reflecting both first-pass success rates and the frequency and effectiveness of strategy revisions, as summarized in Table 3.

As shown in Table 3, the initial strategy yielded moderate success rates (65–75%). However, significant improvements emerged once GPT-4V was prompted for a revision when mistakes were flagged. For task 1 ("medicine bottle grasping", B), the final SR rose

from 0.748 to 0.957, illustrating how a single correction could resolve common oversights (such as slight coordinate misalignments). Similarly, the most challenging task 10 ("Stack Multiple Blocks", HA) demonstrated a notable gain, climbing from 0.657 to 0.881. This underscores the importance of rechecking in scenarios involving intricate object arrangements. Although the secondary verification does not guarantee perfect corrections every time, the data confirm that it considerably enhances overall strategy reliability for all tasks.

Table 3. Strategy layer outcomes with secondary verification.

Task	Initial Strategy SR	Strategy Revisions Initiated	Successful Revisions	Finial Strategy SR
Task 1	0.748	50	44	0.957
Task 4	0.710	58	44	0.919
Task 6	0.652	69	49	0.886
Task 10	0.657	62	47	0.881

3.2.2. Evaluation Layer Performance Evaluation

Once the strategy layer has finalized its plan, whether it has been revised or not, the execution layer carries out each motion step. Parallel to this, an evaluation layer monitors correctness at periodic step intervals, governed by an evaluation factor α that determines how frequently each step is checked. In our implementation, α took values of 0, 0.5, and 1.

- $\alpha = 0$, checks only when the entire task finishes;
- $\alpha = 0.5$, checks approximately half of the steps;
- $\alpha = 1$, checks every step in real time.

We ran each task under all three α settings (70 trials each) and recorded the final success rates, execution durations, and error correction statistics. Table 4 illustrates representative results. Each row corresponds to the same tasks from Table 3, tested with various evaluation factor settings.

Table 4. Execution accuracy and duration under different evaluation layer configurations.

Task	α	Error Step Count	Corrected Step Count	Final SR	Avg. Time (s)
Task 1	0	1	1	0.886	34
	0.5	2	1	0.871	35
	1	1	0	0.914	33
Task 4	0	3	1	0.771	35
	0.5	6	2	0.857	56
	1	8	3	0.843	53
Task 6	0	4	2	0.714	59
	0.5	9	2	0.757	82
	1	12	5	0.800	110
Task 10	0	5	1	0.686	69
	0.5	11	4	0.714	88
	1	10	4	0.771	106

Although detailed step-by-step verification ($\alpha = 1$) often yields a slightly higher success rate, it can significantly prolong execution time, as seen with the "T-shaped block arrangement" (A) task (59 s vs. 110 s). By contrast, sparse checking ($\alpha = 0$) is faster but may fail to detect mid-task anomalies until completion, leading to more rework or unrecognized failures. In essence, selecting α involves balancing speed and thoroughness. For relatively simpler tasks (B and I), continuous validation does not drastically alter success rates, whereas advanced tasks (A and HA) benefit more from frequent checks but suffer in efficiency.

3.2.3. Comparison of the RTPF with State-of-the-Art Frameworks and Different VLMs

The proposed GPTArm framework advances robotic task planning and execution in dynamic and unstructured environments. Table 5 compares the SR of VLM-integrated robotic arms in sparse environments for single-object grasping tasks.

Table 5. Comparative performance of VLM-driven robotic grasping systems in sparse single-object grasping.

Reference (Year)	Model	SR	Generalization	Instruction Input
GPTArm (This Work)	GPT-4V	0.900	Unseen	Image, Speech, and Text
[28]-(2024)	Robot-GPT	0.800	Unseen	Text
[44]-(2024)	Qwen-vl	0.767	Unseen	Image and Text
[29]-(2023)	PaLM-E	0.740	Unseen	Image and Text
[43]-(2024)	GPT-4	0.913	Seen	Text
[31]-(2024)	GPT-4V	0.900	Seen	Image and Text
[14]-(2024)	GPT-4V	0.833	Seen	Image and Text

As highlighted in Table 5, GPTArm achieves an SR of 0.900, outperforming recent benchmarks such as Robot-GPT (0.800) [27], Qwen-vl (0.767) [42], and PaLM-E (0.740) [43]. Notably, this performance superiority is particularly evident in handling unseen objects, where GPTArm leverages GPT-4V's zero-shot reasoning and adaptive verification mechanisms. Two key innovations contribute to this success: (1) the hierarchical RTPF architecture, which decouples perception, planning, and execution while enabling iterative refinement through closed-loop feedback, and (2) GPT-4V's ability to interpret user intent and generate context-aware strategies, even under partial occlusions or ambiguous instructions.

To evaluate how well the RTPF accommodates different VLMs, we performed repeated tasks with Fuyu-8B, Qwen-vl-plus, and GPT-4V. Each task targeted a distinct level of complexity, ranging from straightforward single-object manipulation to more elaborate multi-step arrangements. Figure 4 presents the bar charts for SR, GCR, and EXEC, enabling direct comparisons among the three models under identical conditions.

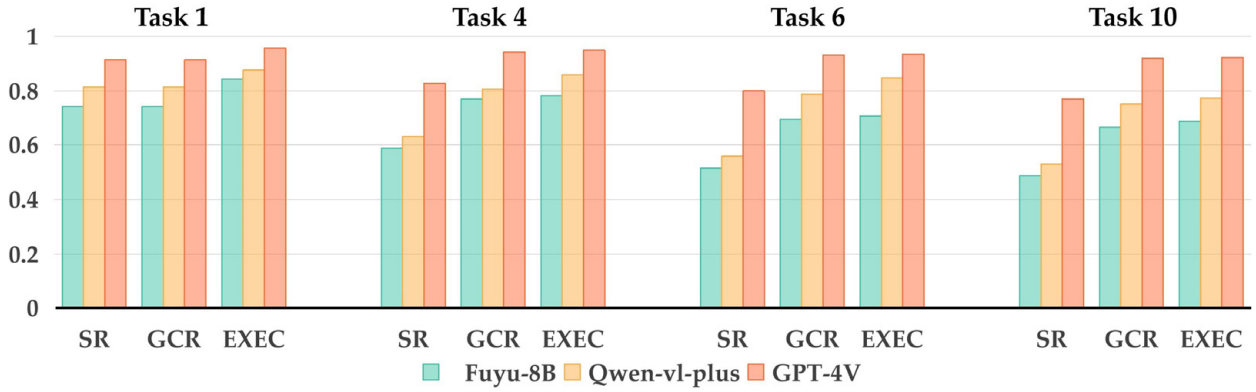

Figure 4. Comparative performance of Fuyu-8B, Qwen-vl-plus, and GPT-4V across four representative tasks.

The bar charts indicate that GPT-4V maintains a notable performance advantage over the other two models, particularly in complex tasks such as 6 and 10. For example, in Experiment 6 (additional objects/repositioning), GPT-4V achieves an SR of 0.800, a GCR of 0.932, and an EXEC of 0.935, while Fuyu-8B records 0.514, 0.696, and 0.708, respectively, and Qwen-vl-plus lies in between (0.557, 0.789, 0.848). Even in simple tasks (e.g., Experiment 1), GPT-4V shows an SR of 0.914 compared to Fuyu-8B at 0.743 and Qwen-vl-plus at 0.814, demonstrating its consistent superiority across varying task complexities.

While GPT-4V's results appeared strongest, Fuyu-8B and Qwen-vl-plus also showed acceptable performance, indicating the RTPF's flexibility with different VLM architectures. This adaptability suggests the potential for extending the framework to other VLMs, enabling broader application scenarios.

3.3. The Performance Evaluation of GPTArm Across Multiple Tasks

3.3.1. Overview of Completed Task Set

To investigate the practical performance of GPTArm, we conducted ten distinct experiments using GPT-4V as the underlying VLM. Introduced in Section 3.1, these experiments encompass a range of manipulation tasks, each requiring varying degrees of object recognition, grasp precision, and multi-step planning. To test GPTArm's capacity for generalization, we grouped the ten tasks into two categories:

- Seen (Tasks 1, 2, 4, 5, and 7): The target objects for these tasks were present in the YOLOv10 training set, so the system had prior visual knowledge of each item.
- Unseen (Tasks 3, 6, 8, 9, and 10): The target objects for these tasks never appeared in the YOLOv10 training data, posing a more substantial challenge that requires GPTArm to adapt to novel shapes, colors, or configurations.

All ten experiments used $\alpha = 0$ in the evaluation layer. Table 6 presents success rates for each task based on 70 repeated trials. Despite varying task complexity and object layouts, GPTArm maintains robust performance. The highest success rates appear in tasks where the target objects were already "seen" by YOLOv10, yet the system also shows strong potential with "unseen" items. This result highlights GPTArm's adaptability when confronted with shapes or configurations outside of the training dataset.

Table 6. Performance of GPTArm (using GPT-4V) across ten real-world tasks.

Task	Number of Successes/Total Attempts	SR $\pm$ Uncertainty
Task 1	64/70	91.4 $\pm$ 3.35%
Task 2	61/70	87.1 $\pm$ 4.01%
Task 3	63/70	90.0 $\pm$ 3.58%
Task 4	59/70	84.3 $\pm$ 4.35%
Task 5	62/70	88.6 $\pm$ 3.80%
Task 6	56/70	80.0 $\pm$ 4.78%
Task 7	59/70	84.3 $\pm$ 4.35%
Task 8	52/70	74.3 $\pm$ 5.22%
Task 9	55/70	78.6 $\pm$ 4.90%
Task 10	54/70	77.1 $\pm$ 5.02%

Table 5 presents the success rates (SRs) of GPTArm across ten real-world tasks, ranging from 74.3% (Task 8) to 91.4% (Task 1). GPTArm demonstrates notable strengths, particularly in tasks such as Task 1 (91.4%), Task 3 (90.0%), and Task 5 (88.6%), where it achieves consistently high performance. These results underscore the system's capability to autonomously plan and execute tasks with precision, especially in scenarios with clear objectives and moderate complexity. This feasibility in autonomous task processing highlights GPTArm's potential as a reliable framework for real-world robotic applications.

Despite these strengths, limitations surface in tasks with lower SRs, such as Task 8 (74.3%) and Task 10 (77.1%), reflecting challenges with instruction ambiguity and spatial complexity. These shortcomings indicate that GPTArm's performance can degrade when faced with nuanced task demands or complex contexts, revealing areas where its current configuration falls short of optimal reliability.

To address these issues, we propose targeted enhancements drawing on recent insights. First, crafting more precise task prompts with structured language could mitigate misinter-

pretations, as Liu et al. [43] showed improved model accuracy with refined instructions. Second, enhancing spatial analysis via motion library calibration, precision manipulator upgrades, and depth camera adoption could improve spatial performance [51]. Third, adjusting GPT-4V's training to emphasize diverse spatial scenarios may strengthen its adaptability, a principle supported by Brohan et al. [29]. These specific refinements aim to elevate GPTArm's consistency while building on its established strengths.

3.3.2. Illustrative Example of Autonomous Task Planning

To showcase GPTArm's autonomous planning capabilities, we focus on Task 9, an unseen scenario. The robotic arm must assemble five blocks into a T-shaped structure and place extra blocks on top of the T's center. The principal challenge lies in autonomously generating a multi-step plan that positions multiple blocks in the correct arrangement without manual intervention.

We introduced several task-specific fine-tuning instructions to reduce collisions, overlapping, and related issues. The system also accounts for the gripper's size and object footprints. When a target position is already occupied, the plan relocates the interfering block to a different location before placing the new one.

In our experiments, both voice and text inputs were tested for issuing commands. Figure 5 illustrates a voice-driven run in which the system interprets the spoken request to assemble the T-shape. The strategy first moves a yellow block out of the central workspace. It then detects that the pink block's target coordinates overlap with a green block, prompting a temporary shift of the green block. Once that space is free, the pink block is placed at the T's center, followed by the remaining two green blocks and a final blue block on top.

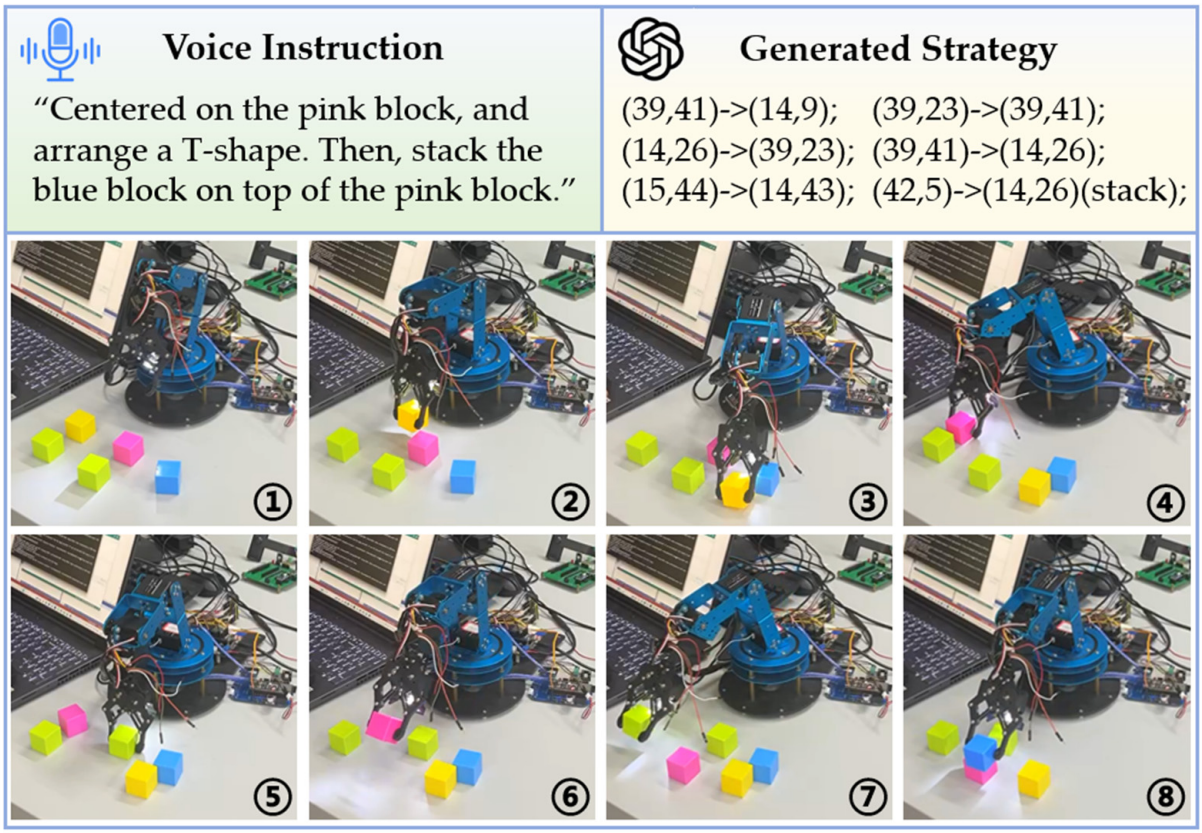

Figure 5. The demonstration of autonomous T-shape assembly and block stacking (Task 9). (1) Enter the initial state. (2) Grasp the yellow block. (3) Place the yellow block. (4) Move the pink block away. (5) Place the green block in the middle. (6) Place the pink block. (7) Place the green block on the side. (8) Stack the blue block.

Although the operation succeeds, GPT-4V does not minimize the total steps. Alternate strategies, such as swapping two blocks earlier, could reduce repositioning. Repeated trials indicate that GPTArm often clears peripheral areas first, possibly due to caution about clustering objects near the center. With more refined prompt engineering and enhanced mechanical precision, one might expect more efficient action sequences. Nonetheless, the system's ability to manage a multi-step routine and reassign objects underscores its strength in autonomous task planning.

3.3.3. Non-Customized Interaction and Dynamic Re-Planning

For ambiguous instruction scenarios, we highlight Task 7, where the system reorganizes objects based on vague user commands. GPT-4V parses natural-language directives that often lack direct item references (for example, "I feel a bit unwell", which the system interprets as a need for medication). By analyzing these colloquial cues semantically, GPTArm can execute the required grasp action even without explicit item names.

Initially, the system places the requested item (a medicine bottle in this task) at the center of the workspace. If the user revises the request, such as "I actually need the charger", the GPTArm recognizes a changed goal. The strategy first puts the bottle back in its original position, then positions the charger in its place. It effectively re-plans the sequence without requiring a special script for each revised request. This layered structure, combined with contextual understanding, enables GPTArm to adapt fluidly to altered instructions. The execution process of this task is presented in Figure 6.

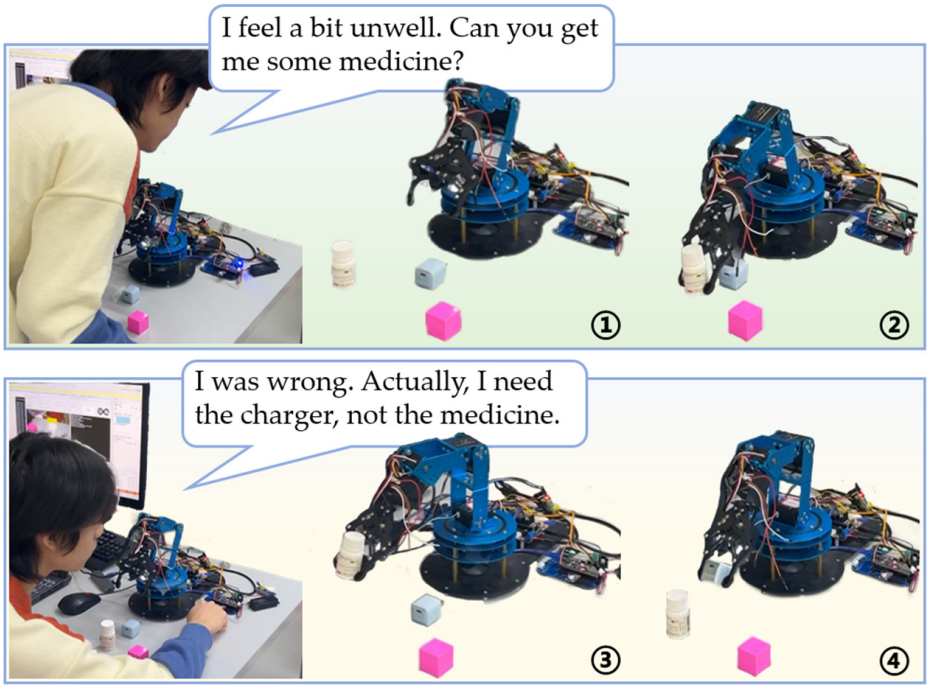

Figure 6. The demonstration of non-customized voice interaction and dynamic re-planning (Task 7). (1) Enter the initial state. (2) Place the medicine bottle in the front. (3) Put the medicine bottle back. (4) Place the charger in the front.

Repeated trials show that even with ambiguous or colloquial directives, GPTArm can interpret user intent and redirect actions as needed, although occasional misidentifications may arise when multiple objects share similar features. Overall, the system's ability to interpret non-custom instructions, accommodate evolving user demands, and adjust its motion plan on the fly suggests a level of adaptability beyond that of a purely preprogrammed mechanism.

These real-world experiments underscore GPTArm's strengths in autonomous strategy generation, multi-step sequencing, and responsive user interaction under uncertain conditions. Whether encountering previously unseen objects or adapting to sudden goal shifts, GPT-4V's integration with the RTPF planning structure provides a robust foundation for sophisticated manipulative tasks.

4. Discussion

The development of GPTArm and its underpinning robotic task processing framework (RTPF) marks a significant advancement in autonomous robotic manipulation, particularly in bridging the semantic and operational gaps that have long challenged traditional robotic systems. The experimental results across ten diverse tasks (Section 3.3, Table 6) demonstrate GPTArm's efficacy, with success rates reaching up to 91.4% in standardized benchmarks (Task 1). This performance underscores the system's proficiency in autonomous task planning, dynamic error recovery, and multimodal interaction—core innovations of the RTPF. By leveraging GPT-4V's reasoning capabilities alongside YOLOv10's precise object localization, GPTArm achieves robust adaptability, excelling in both seen and unseen object scenarios (e.g., Tasks 3 and 6, with SRs of 90.0% and 80.0%, respectively). These outcomes validate the RTPF's hierarchical approach, which decouples perception, planning, and execution while integrating real-time feedback loops to ensure resilience in dynamic environments.

When compared to state-of-the-art frameworks, GPTArm offers distinct advantages. For instance, Robot-GPT [28] achieves an SR of 0.800 for single-object grasping but lacks the closed-loop error recovery mechanisms that enable GPTArm to self-correct mid-task (Section 3.2.2). Similarly, PaLM-E [29] (SR 0.740) and Qwen-vl [44] (SR 0.767) demonstrate versatility in multimodal inputs but fall short in handling unseen objects with the same consistency as GPTArm (Section 3.2.3, Table 5). The RTPF's hierarchical decomposition and secondary verification process (Section 2.1.2) address limitations in prior works, such as the absence of dynamic re-planning in [43] or the reliance on pre-trained datasets in [31]. This positions GPTArm as a forward step in VLM-driven robotics, particularly for tasks requiring contextual reasoning and zero-shot generalization.

GPTArm's natural language interface democratizes robotic control, enabling non-experts to use complex systems for applications like assistive healthcare (e.g., fetching medication) and smart home automation (e.g., object organization). Its modular, software-centric design (Section 2.1.5) ensures scalability across diverse robotic platforms, lowering costs compared to hardware-intensive solutions like stereo vision systems [39], and enhances industrial automation efficiency in dynamic settings.

However, the system is not without constraints. The reliance on GPT-4V's API introduces latency, averaging 33–110 s per task (Section 3.2.2, Table 4), which may limit real-time applicability in time-critical scenarios. Additionally, while effective for rigid objects, GPTArm struggles with irregularly shaped or deformable items (e.g., Task 10, SR 77.1%), where geometric reasoning remains suboptimal due to the 2D coordinate-based operation (Section 2.1.5). These limitations indicate the RTPF's current design falls short of fully addressing the complexities of unstructured environments. Future improvements include reducing API latency via local deployment or streamlined protocols for real-time use, enhancing zero-shot recognition of irregular objects with few-shot learning or physics-informed models [34], and extending the RTPF to multi-arm coordination and deformable object manipulation using federated learning and physics-based planning to broaden its real-world applicability.

5. Conclusions

This study presents GPTArm, an AI-driven robotic arm system that integrates GPT-4V's multimodal reasoning with adaptive task planning to address real-world manipulation challenges. The RTPF's hierarchical architecture enables autonomous task decomposition, closed-loop error recovery, and environment-aware execution, overcoming the rigidity of traditional pre-programmed systems. Empirical validation across ten tasks demonstrates the framework's efficacy, achieving a peak success rate of 91.4% in standardized scenarios and robust generalization to unseen objects. The system's natural language interface enables non-experts to intuitively control complex operations through voice or text commands, bridging the gap between human intent and robotic execution.

By advancing the synergy of VLMs and robotic control, this work provides a scalable solution for dynamic environments, with potential applications in industrial automation, assistive healthcare, and smart home systems. Future research will focus on latency optimization for real-time deployment and enhanced zero-shot recognition for irregular objects, further pushing the boundaries of autonomous robotic systems.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/machines13030247/s1, Video S1: The performance of GPTArm across multiple tasks.

Author Contributions: Conceptualization, H.W., J.Z. and Z.W.; methodology, H.W. and J.L.; software, J.Z. and J.L.; validation, H.W.; formal analysis, H.W.; investigation, J.Z., Z.W. and J.L.; writing—original draft preparation, J.Z. and Z.W.; writing—review and editing, J.Z. and H.W.; visualization, J.Z.; supervision, H.W.; project administration, H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in this article. Further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

VLMs	vision–language models
LLMs	large language models
RTPF	robotic task processing framework
SR	success rate
EXEC	executability
GCR	Goal Condition Recall
TGC	Total Goal Condition
UGC	Unmet Goal Condition
DoF	degrees of freedom
IK	inverse kinematics

Appendix A

Appendix A.1. YOLOv10 Performance Analysis

In this project, we trained a YOLOv10 model on various objects using an NVIDIA RTX 2080Ti GPU. After training, YOLOv10 automatically performs validation and generates relevant evaluation metrics. Precision and recall are emphasized, as their trends directly reflect the model's training effectiveness. In general, smaller fluctuations in these two metrics

indicate better convergence and more stable performance. Figure A1 presents the precision and recall curves throughout the training epochs.

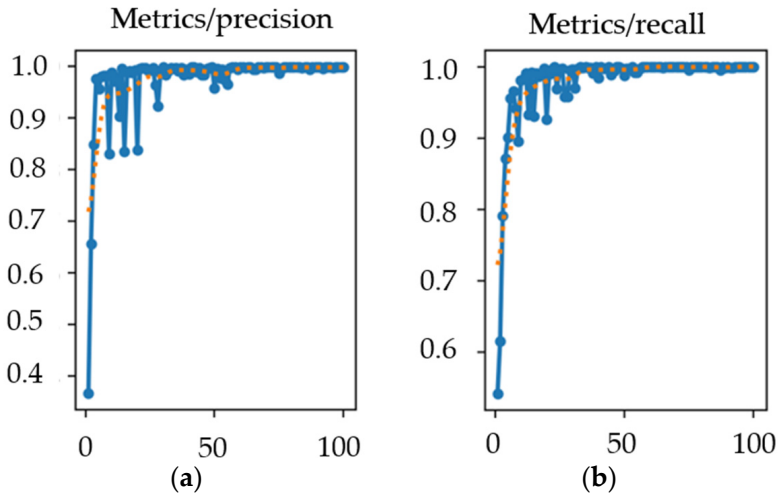

Figure A1. (**a**) Line chart of the change in model accuracy with the number of training rounds. (**b**) Line chart of the change in the model's recall rate with the number of training rounds.

The F_1 score, an important evaluation metric in classification problems, combines precision and recall to assess model performance, and it is often used as the final evaluation metric in machine learning competitions for multi-class classification problems. Here, we introduce the F_1 score to evaluate model performance. The score ranges from 0 to 1, with values closer to 1 indicating better performance. The calculation formula for the F_1 score is as follows:

$$F_1 = \frac{2 \times piecision \times recall}{piecision + recall}, \tag{A1}$$

The model achieved the best performance at the 73rd training epoch, with a precision of 0.99913 and a recall of 1. According to Formula (A1), the F_1 score was calculated to be 0.99956, a result very close to 1. This indicates that the trained model has excellent classification ability for the experimental target objects and meets the project requirements.

Figure A2a,b are the Precision–Confidence curve and the Confusion Matrix obtained during training, respectively.

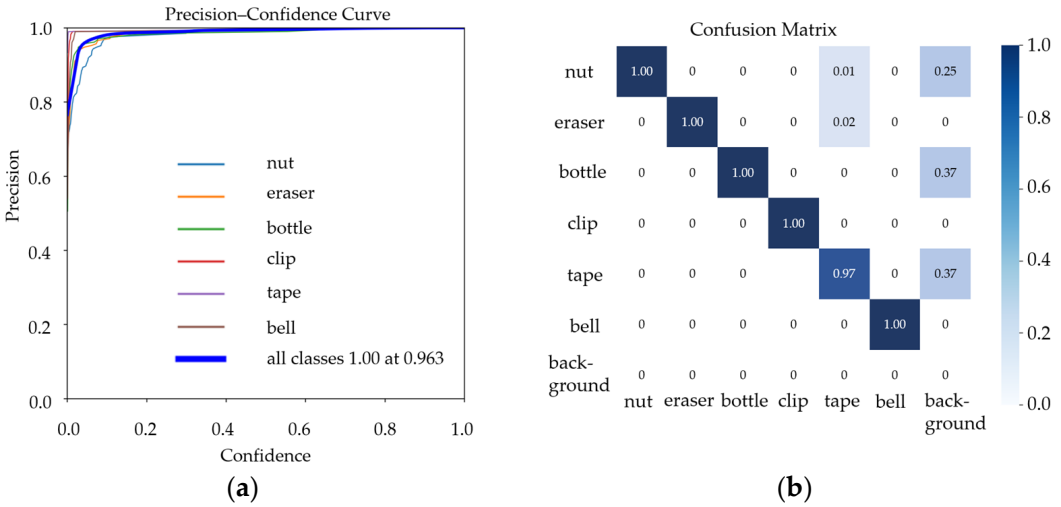

Figure A2. (**a**) Precision–Confidence curve; (**b**) Confusion Matrix.

In Figure A2a, we can see that the overall Precision–Confidence curve is relatively smooth, indicating that the model's precision varies little across different confidence thresholds. This suggests stable performance and high detection reliability. Furthermore, the Precision–Confidence curves for all classes approach 1.0, meaning the model maintains high precision even as the confidence threshold increases, resulting in few false detections.

In Figure A2b, it is evident that the model achieves high true positives (TPs) and true negatives (TNs) across multiple classes, while false positives (FPs) and false negatives (FNs) remain low. This demonstrates the model's strong recognition accuracy, with minimal false positives and negatives. However, some classes—such as the "background" category—still exhibit a certain misclassification rate, which may require further dataset optimization or hyperparameter tuning to improve the model's discrimination ability for these classes.

In conclusion, the smooth Precision–Confidence curve and balanced Confusion Matrix both validate the model's stability and efficiency. They show that YOLOv10 performs well in this task, accurately identifying different classes while keeping false detections and omissions to a minimum, thereby meeting practical application requirements.

Appendix A.2. Robotic Arm Grasping Algorithm

Appendix A.2.1. Overview of the Robotic Arm Grasping Task

This system employs a 6-DoF small robotic arm for target grasping tasks. The grasping process consists of four core steps: target detection, coordinate transformation, motion planning, and executing the grasp. YOLOv10 is used for target recognition, and in combination with the robotic arm's kinematics to calculate joint angles, it enables the end-effector to accurately grasp the target.

Appendix A.2.2. Target Detection and Grasping Strategy

YOLOv10 is employed to recognize objects, providing bounding box coordinates (x, y, w, h) along with their category labels.

In this project, the TOF distance measurement module selected is the VL53L1X. The TOF module's distance measurement mode is set to short-range, with a measurement range of 40–1300 mm and an accuracy of $\pm$25 mm.

Since the system also uses a 2D camera for environmental data, we combined the TOF readings with the bounding box width to decide whether an object is within grasping range. Because our camera is fixed beneath the gripper, when an object is detected, the system determines the approximate distance of the object from the robotic arm based on the object's type and the width of its detection box. If the detection box width is small—indicating that the object occupies a small area in the camera view—the system infers that the object is too far from the gripper to be grasped and will command the robotic arm to move closer. Conversely, if the detection box width is large and the TOF module confirms that the object is within a graspable range, the system transitions from merely approaching the object to executing a grasp, thereby implementing a reliable grasping strategy.

Below is the calculation method we used:

According to the principles of perspective projection, the estimated distance Z from the target to the camera is inversely proportional to the width A of its detection bounding box in the image. This relationship can be approximated by the formula

$$Z \approx \frac{K}{A}, \tag{A2}$$

where Z is the estimated distance from the target to the camera. A is the width of the target's detection bounding box in the image, and K is a calibration factor determined

through experiments; its specific value depends on the camera's focal length, the imaging sensor size, and the actual size of the target.

Once we obtain Z, we can combine it with the target's category (which informs us of its actual size) and the distance measured by the TOF module from the object to the gripper to determine whether the object is within grasping range. When the distance requirement is met, knowing the target's category allows us to adjust the gripper's force to prevent the object from being damaged or grasped unstably.

For ease of subsequent use, we designed a GUI interface primarily developed using Qt. Its robust code encapsulation mechanism ensures high modularity, enabling rapid development while maintaining an aesthetically pleasing design. The GUI adopts a modular design, consisting mainly of a function control module, a system control module, and a graphical processing and display module. The GUI's operation is shown in Figure A3.

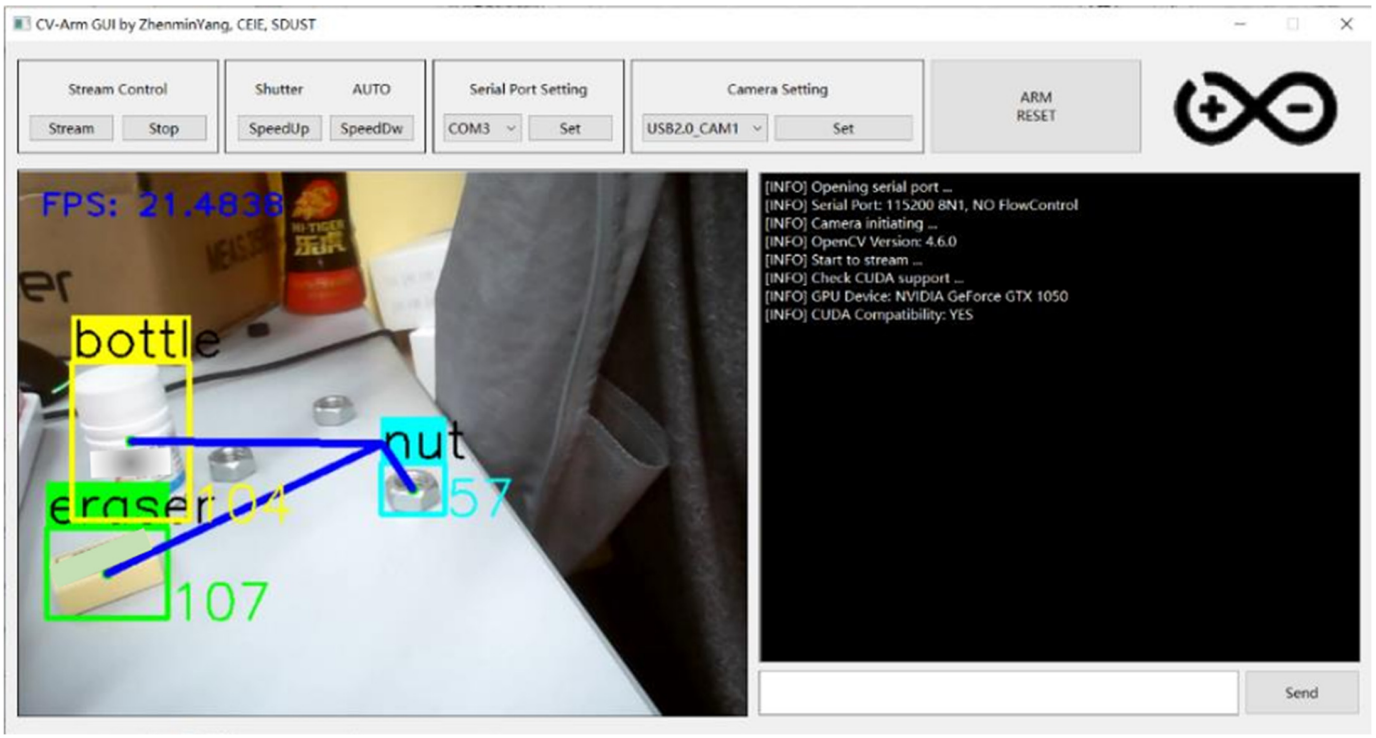

Figure A3. Operating effect of GUI.

In Figure A3, we can see that each detection bounding box is accompanied by a number, which represents the width of the bounding box. Additionally, a purple line connects the center of the detection box to the center of the image; this line indicates the offset of the target relative to the center point of the image. By analyzing the offset data, we can determine the target's offset relative to the robotic arm.

Appendix A.2.3. Offset Calculation

After detecting the target, the system needs to convert the pixel coordinates into a world coordinate system usable by the robotic arm to calculate the target's offset relative to it.

Conversion from Pixel Coordinates to Camera Coordinates

The target's center pixel coordinates (u, v) provided by YOLOv10 need to be converted into the camera coordinate system (X_C, Y_C, Z_C), and this conversion is calculated using the camera's intrinsic parameter matrix (focal lengths f_x, f_y, and principal point offsets c_x, c_y).

$$X_C = \frac{(u - c_x)Z_C}{f_x}, \quad Y_C = \frac{(v - c_y)Z_C}{f_y}, \quad Z_C = Z, \tag{A3}$$

Conversion from Camera Coordinates to World Coordinates

Hand–eye calibration is used to obtain the transformation matrix $T_{cam2world}$ from the camera coordinate system to the world coordinate system (i.e., the robotic arm coordinate system). The homogeneous transformation formula is used:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = T_{cam2world} \times \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}, \tag{A4}$$

Calculate the Target's Offset Relative to the Robotic Arm

Let the robotic arm's base be located at (X_0, Y_0, Z_0); then, the target's offset relative to the arm is given by:

$$\Delta X = X_w - X_0, \ \Delta Y = Y_w - Y_0, \ \Delta Z = Z_w - Z_0, \tag{A5}$$

Through these calculation formulas, we can convert the distance from the object's center in the image to the image center into the direction and magnitude by which the robotic arm should move. Once this offset is determined, the position of the robotic arm's end-effector—that is, the target point P (x_0, y_0)—can be computed. With the target point P known, the robotic arm's kinematics can then be used to calculate the joint angles required for positioning the end-effector at P.

Appendix A.3. Robotic Arm Motion Analysis

Appendix A.3.1. Robotic Arm IK Calculation

We implemented the RTPF on the LeArm robotic arm used in our experiments, developing motion library code capable of reaching specified positions within the camera's field of view. While RTPF offers versatility across robotic arm types, its deployment on different arms requires motion library adjustments based on target arm kinematics.

Our robotic arm exhibits satisfactory motion control for regular-shaped objects like building blocks (Tasks 3, 6, 8, 9, and 10), but performance degrades when handling objects with minimal gripper contact areas (e.g., nuts and cylindrical bottles). To address this, we introduce specific control algorithms for fine-tuning motion when approaching the target object, thereby enhancing the accuracy of grasping such items.

We adopted the analytical method [45] as the kinematics solving algorithm. To reduce computational complexity, we simplified the three-dimensional space into two two-dimensional planes: the Yaw plane and the Pitch plane. The intersection of the robotic arm's Yaw axis drive motor and the lower drive motor of the upper gimbal unit serves as the reference point. The abstract position of the end effector is denoted as point P, with L_1, L_2, and L_3 representing the three segments of the robotic arm's links. By decomposing these segments along the x-axis and y-axis, the coordinates of point P, denoted as (x_0, y_0), can be expressed as ($x_1 + x_2 + x_3$, $y_1 + y_2 + y_3$). In the schematic diagram, the angles θ_1, θ_2, and θ_3 are computed using the IK algorithm.

Based on geometric relationships, the following equations are derived:

$$x_0 = L_1 cos\theta_1 + L_2\, cos(\theta_1 + \theta_2) + L_3\, cos(\theta_1 + \theta_2 + \theta_3), \tag{A6}$$

$$y_0 = L_1 sin\theta_1 + L_2\, sin(\theta_1 + \theta_2) + L_3\, sin(\theta_1 + \theta_2 + \theta_3), \tag{A7}$$

In the above equation, x_0 and y_0 are known quantities, and the link lengths are also known. By substituting the angle between the end-effector and the horizontal plane, $\alpha = \theta_1 + \theta_2 + \theta_3$, and denoting the unknown variables in the above equation as m and n, respectively, we obtain the following equation:

$$m = L_3\, cos(\alpha) - x_0, \tag{A8}$$

$$n = L_3\, sin(\alpha) - y_0, \tag{A9}$$

Substituting m and n into the existing equation and simplifying yields the following equation:

$$L_2 = (L_1 \sin\theta_1 + n)^2 + (L_1 \cos\theta_1 + m)^2, \tag{A10}$$

Based on the quadratic formula and the relationships between trigonometric functions, the solution to the equation is obtained:

$$\sin\theta_1 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \tag{A11}$$

$$a = \left(m^2 + n^2\right), \tag{A12}$$

$$b = \frac{-n\left({L_1}^2 - {L_2}^2 - m^2 - n^2\right)}{L_2}, \tag{A13}$$

$$c = \left(\frac{L_1^2 - L_2^2 - m^2 - n^2}{2L_2}\right)^2, \tag{A14}$$

Similarly, the value of θ_2 can be determined. After obtaining the values of θ_1 and θ_2, the value of θ_3 can be deduced.

These IK equations are derived under the assumption that the target position lies in the first quadrant of the coordinate system (i.e., $x > 0$, $y > 0$). In other quadrants, adjustments may be required to account for negative coordinates, as the current formulation could lead to imaginary solutions or singularities if not adapted accordingly. In our system, predefined motion library code is used to approach coordinate points within the camera's view. Near the points, IK algorithms are employed for precise motion control. Under normal circumstances, this ensures that the object to be grasped remains within the camera's range, thereby avoiding complex scenarios involving other quadrants.

Additionally, the existence of real solutions to the IK equations depends on the target position being within the robot's reachable workspace. For example, if the target is too far or too close, the discriminant of the quadratic equation ($b^2 - 4ac$) may become negative, resulting in no real solutions. This reflects a physical limitation of the robot's reach.

The entire process is illustrated in Figure A4.

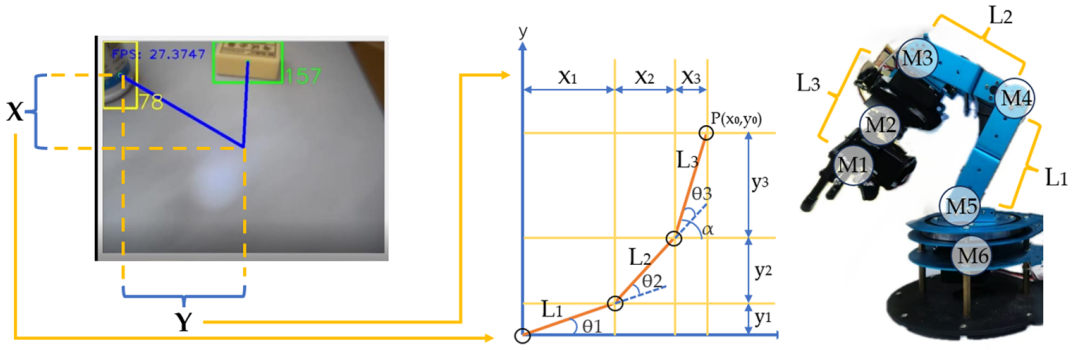

Figure A4. Schematic figure of the robotic arm grasping algorithm.

Then, the value of θ_2 can be determined. Once θ_1 and θ_2 are calculated, θ_3 can be derived. Upon solving for θ_1, θ_2, and θ_3, the target joint angles are obtained.

Appendix A.3.2. PID Control for Adjusting Robotic Arm Motion

Inspired by the previous work on vision-based robotic control [52], we computed the offset between the object's center and the camera's viewport center to derive PID control increments. This algorithm ensures alignment during grasping and tracks slow-moving objects within the field of view. Independent control functions for the Yaw and Pitch axes were developed as below:

- Yaw axis control: Based on the horizontal offset, the PID algorithm computes a speed adjustment applied to the sixth motor, which is responsible for horizontal rotation, to regulate the robotic arm's horizontal angle.
- Pitch axis control: Based on the vertical offset, the PID algorithm calculates a speed adjustment primarily for the third motor, the main vertical rotation axis motor of the robotic arm. When the third motor reaches its limit, the adjustment is transferred to the 5th auxiliary axis motor; otherwise, the fifth motor remains stationary.

When the robotic arm executes a grabbing task, different target coordinates impose varying requirements on the grabbing posture. To address this, we predefined corresponding values. Upon receiving a grabbing command, the robotic arm moves to the target coordinate position according to these preset values. During this phase, motors 1, 2, and 4 perform fixed, preprogrammed actions without requiring real-time adjustments. For instance, when executing a grabbing task at specific coordinates, motor 2 rotates to a predetermined angle to optimize the subsequent grabbing motion. This is because motors 1 and 2 primarily control the gripper's opening width and its deflection angle relative to the plane, without influencing the overall posture of the robotic arm. Meanwhile, the adjustment role of motor 4 on the Pitch axis can, to some extent, be substituted by motors 3 and 5.

As the robotic arm moves toward the target coordinate position based on preset values, all motors except 3, 5, and 6 also rotate to new preset positions. This approach effectively expands the robotic arm's potential working range. It may even resolve scenarios where, when relying solely on motors 3, 5, and 6 for grabbing, the condition $4ac > b^2$ results in no solution—due to the object being too distant from the arm or the object's deflection angle relative to the arm being too large—transforming them into solvable cases.

Upon reaching the designated coordinate position, the system uses distance data from the TOF module to assess whether the arm is in a suitable grabbing position, triggering the YOLOv10 recognition and grabbing function. The camera identifies and locates the target object, relaying offset information to the control system. The robotic arm then employs the PID control algorithm for dynamic adjustments to achieve precise object grabbing. The PID control formula is:

$$u_i(k) = K_p e_i(k) + K_I \sum_{j=0}^{k} e_i(j)\Delta t + K_D \frac{e_i(k) - e_i(k-1)}{\Delta t}, \tag{A15}$$

where

- $e_i(k)$ is the difference between the target angle and the current angle.

$$e_i(t) = \theta_i^{target} - \theta_i^{actual}, \quad i \in \{1, 2, 3\}, \tag{A16}$$

- $e_i(k-1)$ is the difference between the target angle and the angle from the previous step.
- $\sum_{j=0}^{k} e_i(j)$ represents the cumulative sum of the angular differences.

- K_P is the proportional gain, which controls the response speed of the robotic arm.
- K_I is the integral gain, which reduces the steady-state error.
- K_D is the derivative gain, which minimizes overshoot and improves system stability.
- $u_i(k)$ is the output PWM signal value.

Through this control strategy, the PID algorithm computes speed adjustments in real time, precisely tuning the robotic arm's motion, eliminating deviations, and enabling the end effector to accurately reach the target position to complete the grabbing task.

References

1. Yang, Y.; Yu, H.; Lou, X.; Liu, Y.; Choi, C. Attribute-Based Robotic Grasping with Data-Efficient Adaptation. *IEEE Trans. Robot.* **2024**, *40*, 1566–1579. [CrossRef]
2. Yang, X.; Zhou, Z.; Sørensen, J.H.; Christensen, C.B.; Ünalan, M.; Zhang, X. Automation of SME production with a Cobot system powered by learning-based vision. *Robot. Comput.-Integr. Manuf.* **2023**, *83*, 102564. [CrossRef]
3. Ge, Y.; Zhang, S.; Cai, Y.; Lu, T.; Wang, H.; Hui, X.; Wang, S. Ontology based autonomous robot task processing framework. *Front. Neurorobot.* **2024**, *18*, 1401075. [CrossRef] [PubMed]
4. Shanthi, M.D.; Hermans, T. Pick and Place Planning is Better than Pick Planning then Place Planning. *IEEE Robot. Autom. Lett.* **2024**, *9*, 2790–2797. [CrossRef]
5. Hao, Z.; Chen, G.; Huang, Z.; Jia, Q.; Liu, Y.; Yao, Z. Coordinated Transportation of Dual-arm Robot Based on Deep Reinforcement Learning. In Proceedings of the 19th IEEE Conference on Industrial Electronics and Applications (ICIEA 2024), Kristiansand, Norway, 5–8 August 2024.
6. Reddy, A.B.; Mahesh, K.M.; Prabha, M.; Selvan, R.S. Design and implementation of A Bio-Inspired Robot Arm: Machine learning, Robot vision. In Proceedings of the International Conference on New Frontiers in Communication, Automation, Management and Security (ICCAMS 2023), Bangalore, India, 27–28 October 2023.
7. Farag, M.; Abd Ghafar, A.N.; ALSIBAI, M.H. Real-time robotic grasping and localization using deep learning-based object detection technique. In Proceedings of the IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS 2019), Selangor, Malaysia, 29–29 June 2019.
8. Ban, S.; Lee, Y.J.; Yu, K.J.; Chang, J.W.; Kim, J.-H.; Yeo, W.-H. Persistent human–machine interfaces for robotic arm control via gaze and eye direction tracking. *Adv. Intell. Syst.* **2023**, *5*, 2200408. [CrossRef]
9. Li, X.; Liu, L.; Zhang, Z.; Guo, X.; Cui, J. Autonomous Discovery of Robot Structure and Motion Control Through Large Vision Models. In Proceedings of the IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE International Conference on Robotics, Automation and Mechatronics (RAM 2024), Hangzhou, China, 8–11 August 2024.
10. Shi, B.; Cai, H.; Gao, H.; Ou, Y.; Wang, D. The Robot's Understanding of Classification Concepts Based on Large Language Model. In Proceedings of the IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO 2024), Hong Kong, China, 20–22 May 2024.
11. He, H.; Li, Y.; Chen, J.; Guo, Y.; Bi, X.; Dong, E. A Human-Robot Interaction Dual-Arm Robot System for Power Distribution Network. In Proceedings of the China Automation Congress (CAC 2023), Chongqing, China, 17–19 November 2023.
12. Cho, J.; Choi, D.; Park, J.H. Sensorless variable admittance control for human-robot interaction of a dual-arm social robot. *IEEE Access* **2023**, *11*, 69366–69377. [CrossRef]
13. Dimitropoulos, N.; Papalexis, P.; Michalos, G.; Makris, S. Advancing Human-Robot Interaction Using AI—A Large Language Model (LLM) Approach. In Proceedings of the European Symposium on Artificial Intelligence in Manufacturing (ESAIM 2023), Kaiserslautern, Germany, 19 September 2023.
14. Tziafas, G.; Kasaei, H. Towards open-world grasping with large vision-language models. *arXiv* **2024**, arXiv:2406.18722.
15. Mirjalili, R.; Krawez, M.; Silenzi, S.; Blei, Y.; Burgard, W. Lan-grasp: Using large language models for semantic object grasping. *arXiv* **2023**, arXiv:2310.05239.
16. Luo, H.; Guo, Z.; Wu, Z.; Teng, F.; Li, T. Transformer-based vision-language alignment for robot navigation and question answering. *Inf. Fusion* **2024**, *108*, 102351. [CrossRef]
17. Que, H.; Pan, W.; Xu, J.; Luo, H.; Wang, P.; Zhang, L. "Pass the butter": A study on desktop-classic multitasking robotic arm based on advanced YOLOv7 and BERT. *arXiv* **2024**, arXiv:2405.17250.
18. Chen, X.; Yang, J.; He, Z.; Yang, H.; Zhao, Q.; Shi, Y. QwenGrasp: A Usage of Large Vision Language Model for Target-oriented Grasping. *arXiv* **2023**, arXiv:2309.16426.
19. Wang, R.; Yang, Z.; Zhao, Z.; Tong, X.; Hong, Z.; Qian, K. LLM-based Robot Task Planning with Exceptional Handling for General Purpose Service Robots. *arXiv* **2024**, arXiv:2405.15646.
20. Vemprala, S.H.; Bonatti, R.; Bucker, A.; Kapoor, A. Chatgpt for robotics: Design principles and model abilities. *IEEE Access* **2024**, *12*, 55682–55696.

21. Mao, J.W. A Framework for LLM-Based Lifelong Learning in Robot Manipulation. Massachusetts Institute of Technology. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, February 2024.
22. Wang, B.; Zhang, J.; Dong, S.; Fang, I.; Feng, C. Vlm see, robot do: Human demo video to robot action plan via vision language model. *arXiv* **2024**, arXiv:2410.08792.
23. Zhang, Y.; Xin, D.; Yang, M.; Xu, S.; Wang, C. Research on Dual Robotic Arm Path Planning Based on Steering Wheel Sewing Device. In Proceedings of the 6th International Symposium on Autonomous Systems (ISAS 2023), Nanjing, China, 23–25 June 2023.
24. Fu, K.; Dang, X. Light-Weight Convolutional Neural Networks for Generative Robotic Grasping. *IEEE Trans. Ind. Inform.* **2024**, *20*, 6696–6707.
25. Chi, M.; Chang, S.; Guo, Z.; Huang, S.; Li, Z.; Li, J.; Xia, Z.; Zheng, Z.; Ren, Q. Research on Target Recognition and Grasping of Dual-arm Cooperative Mobile Robot Based on Vision. In Proceedings of the International Symposium on Intelligent Robotics and Systems (ISoIRS 2024), Changsha, China, 14–16 June 2024.
26. Ko, D.-K.; Lee, K.-W.; Lee, D.H.; Lim, S.-C. Vision-based interaction force estimation for robot grip motion without tactile/force sensor. *Expert Syst. Appl.* **2023**, *211*, 118441. [CrossRef]
27. Bhat, V.; Kaypak, A.U.; Krishnamurthy, P.; Karri, R.; Khorrami, F. Grounding LLMs For Robot Task Planning Using Closed-loop State Feedback. *arXiv* **2024**, arXiv:2402.08546.
28. Jin, Y.; Li, D.; Yong, A.; Shi, J.; Hao, P.; Sun, F.; Zhang, J.; Fang, B. Robotgpt: Robot manipulation learning from chatgpt. *IEEE Robot. Autom. Lett.* **2024**, *9*, 2543–2550.
29. Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Chen, X.; Choromanski, K.; Ding, T.; Driess, D.; Dubey, A.; Finn, C. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv* **2023**, arXiv:2307.15818.
30. Li, B.; Wu, P.; Abbeel, P.; Malik, J. Interactive task planning with language models. *arXiv* **2023**, arXiv:2310.10645.
31. Mei, A.; Zhu, G.-N.; Zhang, H.; Gan, Z. ReplanVLM: Replanning robotic tasks with visual language models. *IEEE Robot. Autom. Lett.* **2024**, *9*, 10201–10208.
32. Bernardo, R.; Sousa, J.M.; Gonçalves, P. Ontological framework for high-level task replanning for autonomous robotic systems. *Robot. Auton. Syst.* **2025**, *184*, 104861.
33. Osada, M.; Garcia Ricardez, G.A.; Suzuki, Y.; Taniguchi, T. Reflectance estimation for proximity sensing by vision-language models: Utilizing distributional semantics for low-level cognition in robotics. *Adv. Robot.* **2024**, *38*, 1287–1306.
34. Han, Y.; Yu, K.; Batra, R.; Boyd, N.; Mehta, C.; Zhao, T.; She, Y.; Hutchinson, S.; Zhao, Y. Learning generalizable vision-tactile robotic grasping strategy for deformable objects via transformer. *IEEE/ASME Trans. Mechatron.* **2024**, *30*, 554–566.
35. Hofer, M.; Sferrazza, C.; D'Andrea, R. A vision-based sensing approach for a spherical soft robotic arm. *Front. Robot. AI* **2021**, *8*, 630935.
36. Shi, W.; Wang, K.; Zhao, C.; Tian, M. Compliant control of dual-arm robot in an unknown environment. In Proceedings of the 7th International Conference on Control and Robotics Engineering (ICCRE 2022), Beijing, China, 15–17 April 2022.
37. Suphalak, K.; Klanpet, N.; Sikaressakul, N.; Prongnuch, S. Robot Arm Control System via Ethernet with Kinect V2 Camera for use in Hazardous Areas. In Proceedings of the 1st International Conference on Robotics, Engineering, Science, and Technology (RESTCON 2024), Pattaya, Thailand, 16–18 February 2024.
38. Wei, J.; Li, J.; Huang, J.; Pang, Z.; Zhang, K. Visual Obstacle Avoidance Trajectory Control of Intelligent Loading and Unloading Robot Arm Based on Hybrid Interpolation Spline. In Proceedings of the 9th International Symposium on Computer and Information Processing Technology (ISCIPT 2024), Xi'an, China, 24–26 May 2024.
39. Zheng, J.; Chen, L.; Li, Y.; Khan, Y.A.; Lyu, H.; Wu, X. An intelligent robot sorting system by deep learning on RGB-D image. In Proceedings of the 22nd International Symposium INFOTEH-JAHORINA (INFOTEH 2023), East Sarajevo, Bosnia and Herzegovina, 15–17 March 2023.
40. Park, Y.; Son, H.I. Visual Scene Understanding for Efficient Cooperative Control of Agricultural Dual-Arm Robots. In Proceedings of the 24th International Conference on Control, Automation and Systems (ICCAS 2024), Jeju, Republic of Korea, 29 October–1 November 2024.
41. Wu, K.; Chen, L.; Wang, K.; Wu, M.; Pedrycz, W.; Hirota, K. Robotic arm trajectory generation based on emotion and kinematic feature. In Proceedings of the International Power Electronics Conference (IPEC-Himeji 2022-ECCE Asia 2022), Himeji, Japan, 15–19 May 2022.
42. Wake, N.; Kanehira, A.; Sasabuchi, K.; Takamatsu, J.; Ikeuchi, K. Gpt-4v (ision) for robotics: Multimodal task planning from human demonstration. *IEEE Robot. Autom. Lett.* **2024**, *9*, 10567–10574.
43. Liu, H.; Zhu, Y.; Kato, K.; Tsukahara, A.; Kondo, I.; Aoyama, T.; Hasegawa, Y. Enhancing the LLM-Based Robot Manipulation Through Human-Robot Collaboration. *arXiv* **2024**, arXiv:2406.14097.
44. Zhao, W.; Chen, J.; Meng, Z.; Mao, D.; Song, R.; Zhang, W. Vlmpc: Vision-language model predictive control for robotic manipulation. *arXiv* **2024**, arXiv:2407.09829.

45. Muslim, M.A.; Urfin, S.N. Design of geometric-based inverse kinematics for a low cost robotic arm. In Proceedings of the 2014 Electrical Power, Electronics, Communicatons, Control and Informatics Seminar (EECCIS 2014), Malang, Indonesia, 27–28 August 2014.
46. Kariuki, S.; Wanjau, E.; Muchiri, I.; Muguro, J.; Njeri, W.; Sasaki, M. Pick and Place Control of a 3-DOF Robot Manipulator Based on Image and Pattern Recognition. *Machines* **2024**, *12*, 665. [CrossRef]
47. Liu, W.; Wang, S.; Gao, X.; Yang, H. A Tomato Recognition and Rapid Sorting System Based on Improved YOLOv10. *Machines* **2024**, *12*, 689. [CrossRef]
48. Sun, H.; Yao, G.; Zhu, S.; Zhang, L.; Xu, H.; Kong, J. SOD-YOLOv10: Small Object Detection in Remote Sensing Images Based on YOLOv10. *IEEE Geosci. Remote Sens. Lett.* **2025**, *22*, 8000705. [CrossRef]
49. Chu, Y.; Xu, J.; Yang, Q.; Wei, H.; Wei, X.; Guo, Z.; Leng, Y.; Lv, Y.; He, J.; Lin, J. Qwen2-audio technical report. *arXiv* **2024**, arXiv:2407.10759.
50. Chu, Y.; Xu, J.; Zhou, X.; Yang, Q.; Zhang, S.; Yan, Z.; Zhou, C.; Zhou, J. Qwen-audio: Advancing universal audio understanding via unified large-scale audio-language models. *arXiv* **2023**, arXiv:2311.07919.
51. Wang, Z.; Zhou, Z.; Song, J.; Huang, Y.; Shu, Z.; Ma, L. Towards testing and evaluating vision-language-action models for robotic manipulation: An empirical study. *arXiv* **2024**, arXiv:2409.12894.
52. Hiba, S.; Smail, T.; Rachid, S.; Abdellah, C. Vision-Based Robotic Arm Control Algorithm Using Deep Reinforcement Learning for Autonomous Objects Grasping. *Appl. Sci.* **2021**, *11*, 7917. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.