ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

Long-horizon Task Completion with Robotic Arms by Human Instructions

BINGJUN GUO (bingjun3)

QI LONG (qilong2)

QINGRAN WU (qingran3)

YUXI CHEN (yuxi5)

(alphabetically)

Sponsor: Gaoang Wang, Liangjing Yang

TA: Tielong Cai, Tianci Tang

April 14, 2025

1 Introduction

1.1 Problem

The application of robotic arms for complex, long-horizon tasks such as assembling, cooking, and packing is rapidly expanding due to their potential to improve efficiency and reduce human labor. However, executing these multi-step operations consistently remains challenging. Such tasks require robots to reason about the interdependencies between subtasks, adapt to dynamic environmental conditions, and integrate continuous feedback effectively. Current robotic manipulation methods struggle with decomposing and chaining task into manageable actions and maintaining robustness throughout execution. Moreover, many existing robotic systems are limited to predefined scenarios with known object interactions, making them unsuitable for dynamic environments where conditions frequently change. To overcome these limitations and enable robotic arms to autonomously manipulate objects based on real-time feedback, there is a critical need for a comprehensive framework that integrates perception, planning, and acting intelligence effectively.

For example, eat and drink are essential for our daily lives, but it is not easy for everyone. Driven by the real-world problem that people with disabilities such as physical impairments may not be able to serve themselves. Simple actions in cooking, such as packing a sandwich, putting in sauces, moving a pizza into a microwave, may be difficult for them, but easy for just a single robot arm. Considering this, We have chosen cooking as the real-world task for implementing our technique of long-horizon task completion with robotic arms based on human instructions.

1.2 Solution

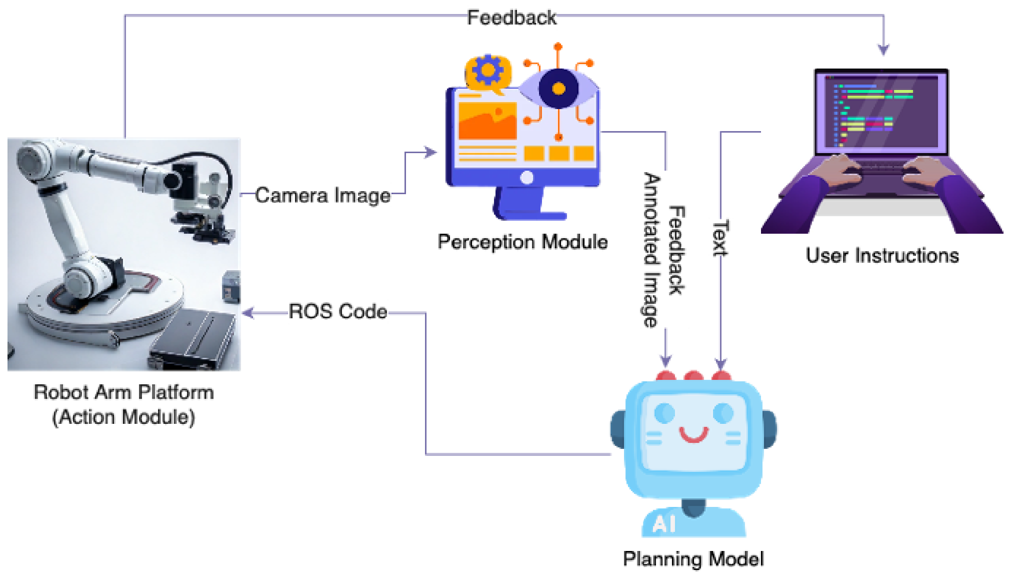


Figure 1: Visual Aid

Our proposed solution addresses the challenges associated with long-horizon robotic manipulation tasks by integrating Perception, Planning, and Acting Intelligence into a cohesive framework. At a high level, our approach leverages advanced sensing technologies alongside intelligent planning algorithms to enable a robotic arm (specifically UR3e) to autonomously execute complex tasks based on human instructions.

In detail, our solution begins with perception: an RGB camera mounted on the top of the scene captures images of the environment and objects involved in the task. Computer vision techniques process these images to accurately identify and localize objects. Next, in the planning stage, we utilize advanced language models capable of interpreting semantic instructions provided by human users alongside processed visual data to generate logical sequences of subtasks. For each subtask, the machine learning model is trained for giving out feasible actions the robot arm should take. Finally, during the acting stage, the robotic arm executes planned movements guided by continuous feedback from visual and tactile sensors integrated into a closed-loop control system.

1.3 High-level Requirements List

1.3.1 Perception Accuracy

The system must achieve at least 90% accuracy in identifying and localizing target objects within its operating environment. Specifically, it should be able to recognize and locate over 20 kinds of common ingredients such as apple, banana, tomato and tools such as plate and pot in our circumstance.

1.3.2 Planning Efficiency

The robot must generate actionable multi-step operation plans within 5 seconds after receiving human instructions. When the plan is being executed, it should be able to modify the plan or next step to align with real world conditions.

1.3.3 Control and Execution Robustness

The robotic arm must successfully complete at least 90% of attempted long-horizon tasks without collisions or critical errors under varying environmental conditions.

2 Design

2.1 Overall Design

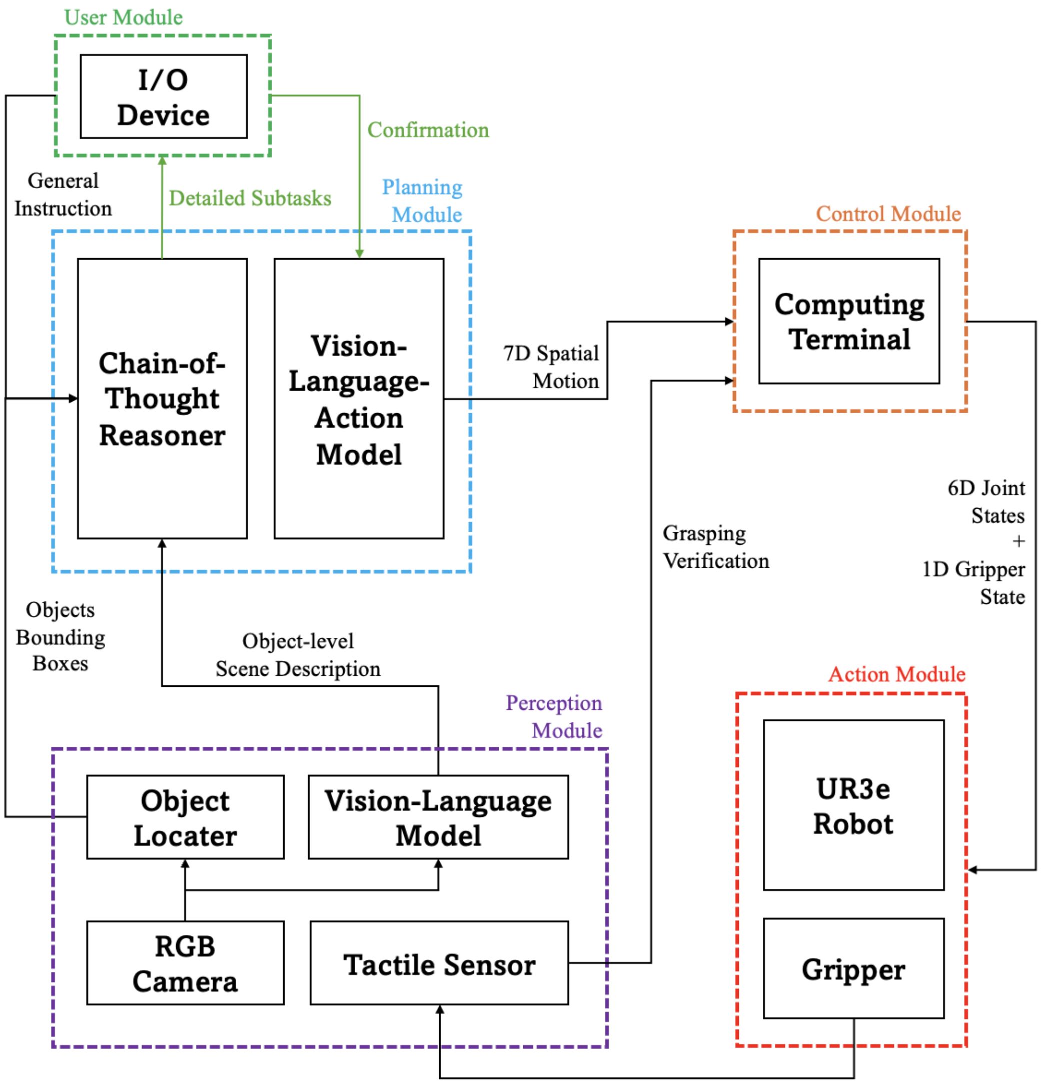


Figure 2: Block Diagram for the Whole System

The overall system consists of five modules, namely User, Perception, Planning, Control, and Action.

2.2 Physical Design

We are using the UR3e robotic arm as our base platform and will design a custom two-finger parallel gripper for object manipulation. The suction-based end effector currently available in our lab is limited to objects with regular shapes or flat surfaces, making it unsuitable for items such as apples and bananas. To overcome this limitation, we plan to build a two-finger gripper driven by a motor and gear sets to handle a wider variety of objects.

A preliminary model of this gripper is shown in the figure below:

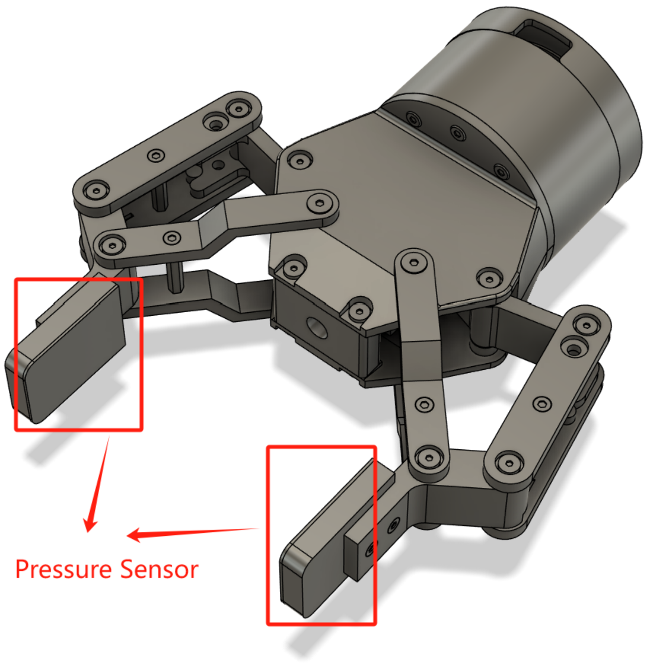


Figure 3: Preliminary Model of Gripper

In this design, we will develop a PCB to control the motor, which in turn actuates the gripper to manipulate objects. The main structure will be 3D printed, with additional parts made from acrylic sheets.

To improve the reliability of object manipulation and ensure that the gripper successfully captures a variety of objects, we plan to use silicone or other high-friction materials in the grippe. In addition, FSR402 force sensors (Figure 4) will be incorporated into the two-finger parallel gripper design. The addition of the FSR402 sensor will provide feedback on whether the gripper has applied adequate force to securely grasp an object.

Figure 4: Force Sensor (FSR402)

The FSR402 sensors will be strategically positioned on the inner surfaces of each gripper finger, where they will make direct contact with the object being manipulated. This placement allows the sensors to detect the force exerted by the gripper when it closes around an object. The sensor outputs an analog signal, which is proportional to the amount of force applied to the object. The signal from the FSR402 will then be transmitted to the control module to confirm whether the object has been properly grasped, the result will be shared with the planning module to make adjustment when needed.

2.3 Subsystems

2.3.1 User Module

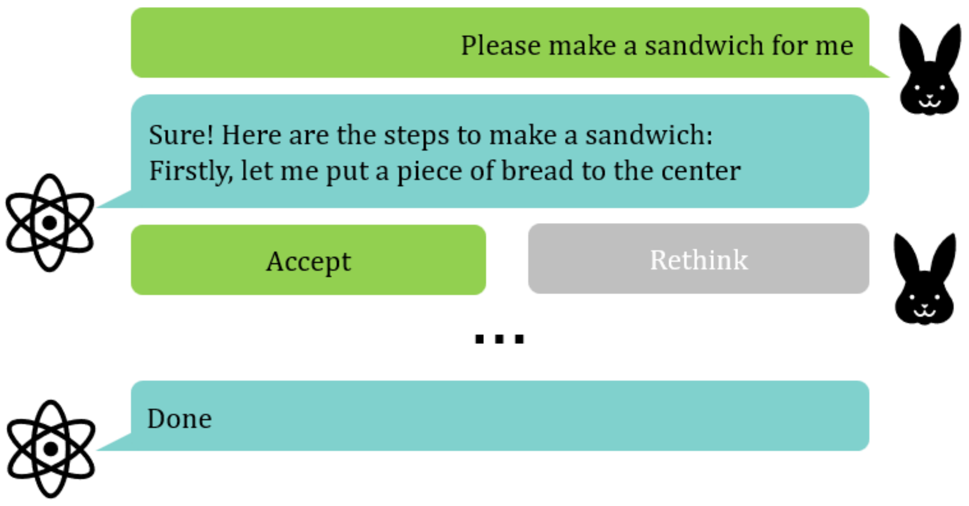

Please make a sandwich for me
Sure! Here are the steps to make a sandwich:
Firstly, let me put a piece of bread to the center
Accept
Rethink
...
Done

Figure 5: User Interface

a) **Overview:** This is the interface of the interaction. The user can input a high-level instruction in natural language using the keyboard, for example, 'Make a sandwich'. The planning process generated from the Planning Module will be output to the User Module and displayed on the computer screen. When the task is finished, a "Done" message will be prompted to the user module and displayed on the computer screen.

b) **Requirements and Verification**

Requirements	**Verification**
The user input should be a high-level task that cannot be solved in less than 10 unit steps.	A. Check by human intuition.
The user input can be received by the Control Module	A. Input an instruction from keyboard. B. Check the input to the Chain-of-Thought Reasoner, the prompt should contain the user instruction.
The thinking process and ”Done” signal can be displayed on computer screen.	A. Input an instruction from keyboard. B. Run our whole system and check as the process goes on, the plan and the final ”Done” signal is displayed on the screen.

2.3.2 Perception Module

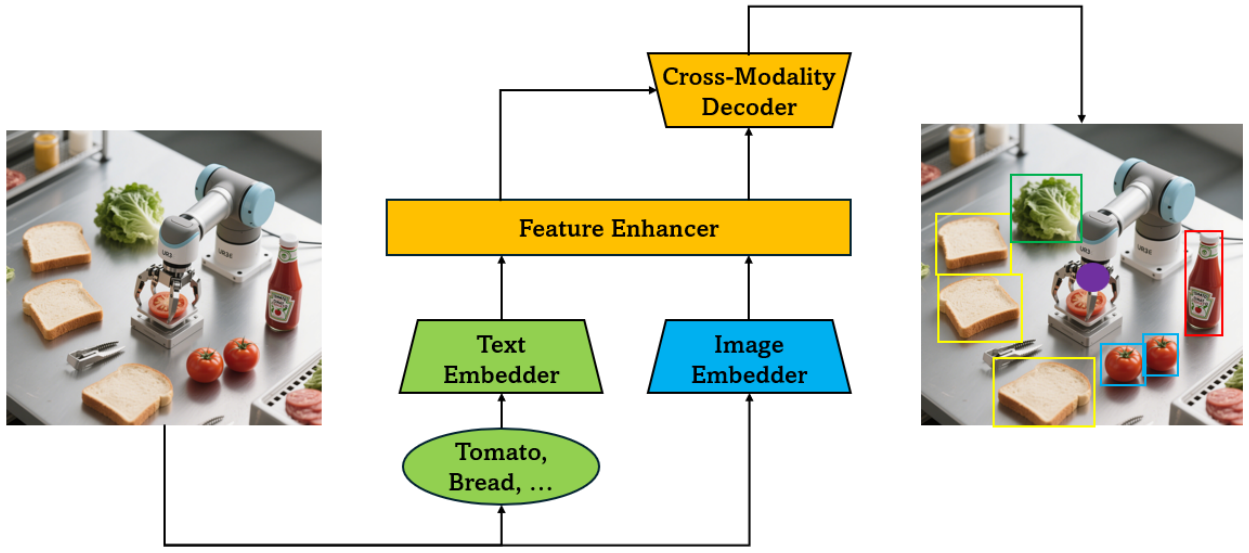


Figure 6: Object Locator

a) **Overview:** It process the signals received from the working environment, which will then be used for planning and acting. It includes two major components: Object Locator, and Sensual Circuit.

b) **Functionalities:** Object Locator is responsible for identifying objects of interest in a scene and labeling them with semantic classification (e.g., a tomato) and region (a bounding box). It is improved based on the Grounding DINO model, which follows a pipeline that firstly embed the text description and the whole image, then use a feature enhancer to learn mutual features and finally decode it in a cross-modality way. The Sensual Circuit is responsible for processing the signal from the force sensor on the gripper into a binary success/fail signal and transmitting it to the planning module.

c) **Workflow:** Requested by the planning module, when a subtask is generated, an image of the scene will be processed by Object Locator, whose annotation result will be sent to VLM (Planning Module) for planning. After each time the robot arm finishes carrying out an action, the snapshot of the scene will be processed by the Object Locator in the same way as above. The sensor circuit will also process the signal from the force sensor on the gripper and send out binary signal to the planner as a feedback.

d) **Requirements and Verification**

Requirements	**Verification**
The input should be supported by a RGB camera.	A. Check the Object Locator's input contains image from RGB camera.
The input should be supported by a force sensor.	A. Check the Object Locator's input contains signal from force sensor.
Object Locator needs to be able to identify at least 20 categories of cooking stuff, including ingredients and cookers, finishing a single computation within 5 seconds.	A. Find an open source object detection test dataset which includes 20 categories of cooking stuff and 400 images in total. B. Run the Object Locator model on this test dataset, verify that it takes less than 2000 seconds in total. C. Check that the Top-1 accuracy of the result should be more than 80 percent.
The force sensor should be able to detect a force range between 0 and 10 N with a resolution of 0.1 N.	A. Test the force sensor with known weights within the specified range and check if the readings match the expected force values within the accuracy tolerance.

2.3.3 Planning Module

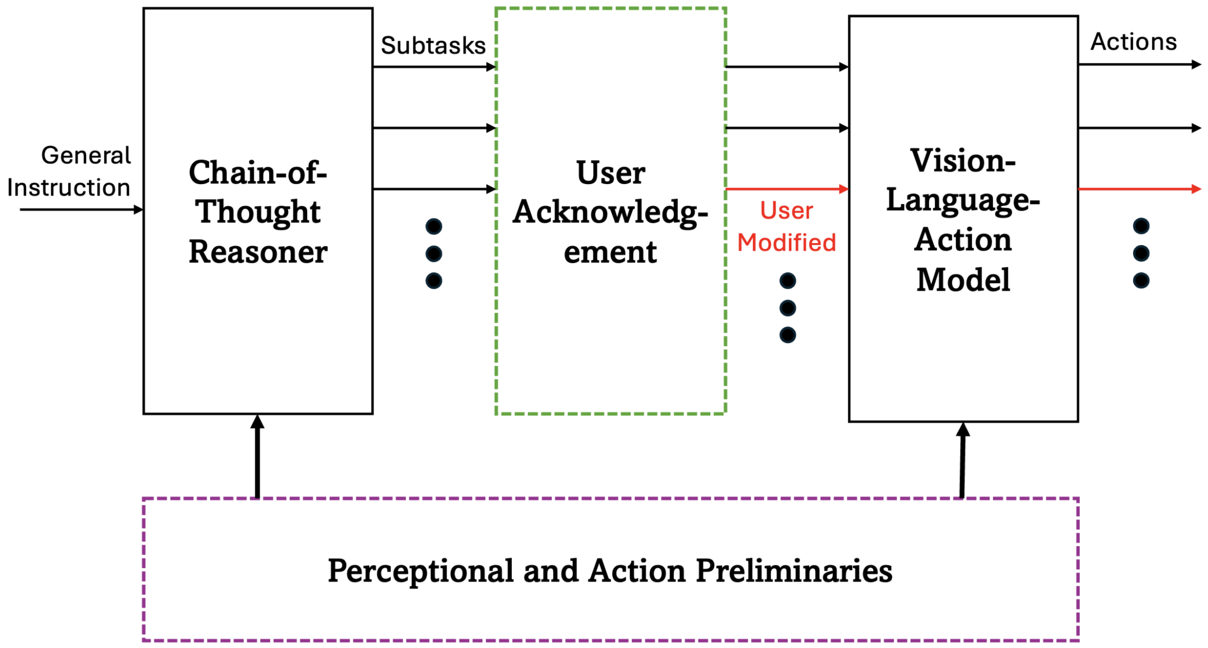

General Instruction
Chain-of-Thought Reasoner
Subtasks
User Acknowledgement
User Modified
Vision-Language-Action Model
Actions
Perceptional and Action Preliminaries

Figure 7: Planning Module

a) **Overview:** The Planning Module is responsible for interpreting user-provided high-level instructions and decomposing them into detailed, actionable subtasks. It bridges the user interface and the robotic control system by combining reasoning capabilities and vision-language understanding to generate feasible spatial motion plans.

b) **Functionalities:** The Planning Module serves several critical functions. First, it interprets the general instruction received from the I/O Device in the User Module and uses the Chain-of-Thought Reasoner to decompose that instruction into a series of detailed subtasks. This enables the system to transform abstract user goals into step-by-step operations that the robot can perform. Meanwhile, it integrates perceptual input from the environment. It receives object-level scene descriptions from the Perception Module, allowing it to reason about the current state of the environment and the spatial relationships between objects. This information is essential for contextual and goal-directed reasoning. With the subtasks and perceptual data in hand, the Vision-Language-Action Model within the Planning Module generates precise 7D spatial motion plans, which include three-dimensional translations and rotations. These plans define how the robot should move in space to complete each subtask. Finally, the Planning Module also supports a feedback mechanism. It sends the proposed subtasks back to the user via the I/O Device for confirmation, ensuring that the system's interpretation of the instruction aligns with user's intent before any physical actions are executed.

c) **Workflow:** The workflow of the Planning Module begins when a user issues a general instruction through the I/O Device. This instruction is passed to the Chain-of-Thought Reasoner, which analyzes it and breaks it down into a sequence of detailed subtasks that are logically and temporally structured. Simultaneously, the Planning Module gathers scene information from the Perception Module, which provides a detailed understanding of the environment, including object bounding boxes and scene context. Once the instruction has been decomposed and the scene is understood, the Vision-Language-Action Model uses this combined input to generate a 7D spatial motion plan and send it to the Control Module. This plan defines the robot's intended movements in space, including how to reach, grasp, or manipulate objects as specified by the user's instruction. Before execution, the generated subtasks and motion plan are sent back to the user via the I/O Device for confirmation, and once confirmed, the plan is forwarded to the Computing Terminal in the Control Module, which then coordinates the robot's physical actions based on the planned trajectory.

d) **Requirements and Verification**

Requirements	**Verification**
The Planning Module must correctly decompose high-level user instructions into subtasks that are ordered in valid and feasible logic.	A. Unit Testing of Chain-of-Thought Reasoner using predefined instructions and expected decompositions. B. Compare generated subtasks with ground truth annotations in benchmark datasets. C. Human-in-the-loop evaluation, where users assess whether the subtasks match their intent. D. Check for logical coherence and completeness across subtasks (e.g., no missing steps or contradictions).
The module must generate plans that are consistent with the current scene, as described by the Perception Module.	A. Inject errors in object positions and check that the planner reacts or fails gracefully. B. Cross-validate spatial plan with object bounding boxes and affordances (e.g., reachable positions). C. Consistency checks between planned motions and scene descriptions to detect mismatches.

The planner must produce physically executable motion plans for the robot in the form of 7D poses (3D position + 4D quaternion orientation).	A. Run kinematic feasibility tests to ensure each 7D pose is reachable by the robot arm. B. Check against robot's joint limits to ensure motion commands are physically executable.
The module must present planned subtasks to the user and incorporate confirmation before execution.	A. Ensure the system pauses execution until confirmation is explicitly received. B. Track versioning of task plans before and after user confirmation to detect changes. C. Test for rejection handling, ensuring revised plans can be generated upon user feedback.
The Planning Module must effectively interface with upstream (perception) and downstream (control) modules.	A. Interface consistency tests: Check that input/output formats match module expectations. B. Latency measurements to ensure that planning happens in real-time or within acceptable bounds. C. Integration testing with the full pipeline under various scenarios (e.g., occlusions, moving targets).

2.3.4 Action Module

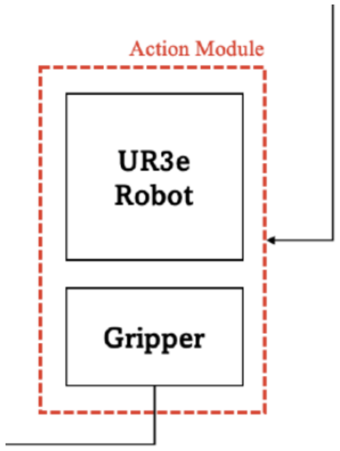
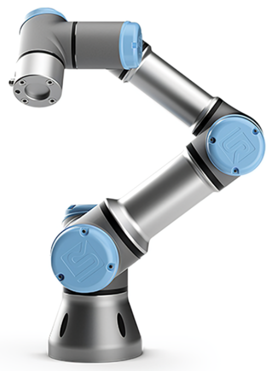


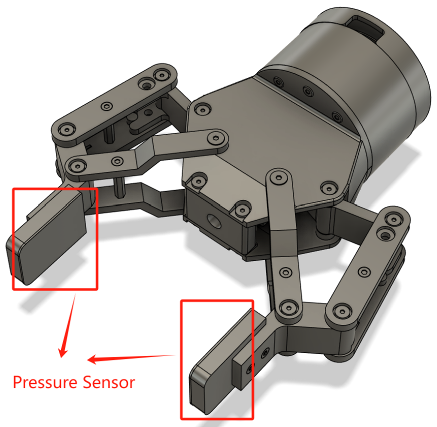
Figure 8: Action Module

a) **Overview:** It interacts with the real environment to execute the tasks. It includes two components: UR3e Robot and a gripper, the end effector, which together enable the robotic system to manipulate objects and complete the task.

b) **Functionalities:** The UR3e robot receives joint movements from the Control Module to reach the desired location, which then enables the mounted gripper to conduct manipulation to the target object. The Optical Sensor captures the distance between gripper and object.

c) **Workflow:** When the Planning Module generates action code, it sends these commands to the Robot Arm, which decode the action via inverse dynamic movement and adjusts its joints and end effector position according to achieve the desired positions and orientations. Specifically, the Planning Module sends these commands to the PCB in the specially designed hand, which controls the motor's motion and subsequently controls the gripper's motion. When the robot finish each single action, the RGB Camera will stream RGB images to the Perception Module for object tracking and the optical sensor will stream gripping signal to the Perception Module for verification. At the start and completion of each sub-goal, the camera captures a snapshot that is processed by the Perception Module for updated object detection, which will be used in the Planning Module.


(a) UR3e Robot



(b) Gripper

d) **Requirements and Verification**

Requirements	**Verification**
The Robot Arm can pick up objects with different sizes (10 mm to 50 mm).	Place three objects with sizes (lengths) 10 mm, 30 mm, 50 mm and test if the Robot Arm can successfully pick them up.
The Robot Arm can pick up objects up to 0.25 kg.	Place an 0.25 kg object and test if the Robot Arm can successfully pick it up.
The Robot Arm can operate safely.	A. Place an object at three different places and test if the Robot Arm can reach it without collision . B. The system can pause immediately for unexpected situations (e.g., people trying to touch the object which is currently grasped by the gripper).

2.3.5 Control Module

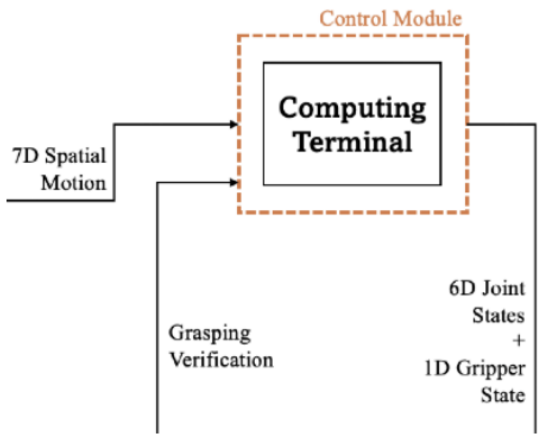


Figure 10: Control Module

a) **Overview:** The Control Module is responsible for managing communication, synchronization, and information exchange between all other modules in our system. It serves as the centralized control terminal, ensuring seamless coordination and execution of tasks. This module is built around a Raspberry Pi, which runs the ROS (Robot Operating System) framework, enabling communication between various sensors, cameras, motors, and model server in real-time.

We selected the Raspberry Pi 4 (Figure 11 as the central processing unit for our smart assistive guide stick due to its robust computational power, extensive interface compatibility, and ability to manage complex logic-based tasks. Powered by a 1.5GHz quad-core Cortex-A72 processor and 4GB of RAM, the Raspberry Pi 4 efficiently handles real-time data processing from the force sensor, motor feedback and camera input. In addition, it supports a variety of communication interfaces, including UART, I2C, SPI, GPIO, and MIPI CSI, ensuring reliable and seamless connectivity with peripheral devices.

b) **Functionalities:** The Control Module has the following key responsibilities:

- Centralized Coordination: It acts as the central hub where all data from the Perception, Planning, Action, and other modules are processed and exchanged. The

Figure 11: Raspberry Pi 4

Raspberry Pi integrates all system components via ROS topics, services, and actions.

- Task Scheduling and Synchronization: The Control Module schedules tasks in the correct sequence and ensures synchronization between modules. For example, it ensures the Perception Module's data is received before the Planning Module can generate the next set of actions.

- Failure Handling: If there is any discrepancy or failure detected by the Verifier or other components, the Control Module can initiate a failure recovery mechanism, such as reprocessing a task or notifying the system for a new plan.

c) **Workflow:** The Control Module is the central communication platform for all system modules. It runs ROS and establishes communication channels for data transfer:

When the Planning Module generates a new task or subtask, the Control Module relays the corresponding ROS command to the Action Module for execution. The Action Module then completes the task (e.g., moves the Robot Arm to a specific position), while the Control Module monitors the process, receives data from the sensors, and informs the Perception Module for real-time feedback. Simultaneously, the Control Module processes the feedback data from the Perception Module to update the task status and make decisions, passing the updated information to the Planning Module for the next step. If a failure occurs (e.g., the object is not detected properly), the Control Module can send recovery instructions to the Action Module or reset the process via the Planning Module.

d) **Requirements and Verification**

Requirements	**Verification**
The system should allow communication between all modules (Perception, Planning, Action, etc.).	Test if the Raspberry Pi 4 can send and receive messages between modules using ROS.
The Raspberry Pi should process data from sensors, motors, and cameras in real-time.	Check if the Raspberry Pi 4 processes input data (e.g., from sensors and cameras) without delay.
The system should handle failures or errors during operation.	Simulate an error (e.g., object not detected) and check if the Control Module initiates recovery actions.

2.4 Tolerance Analysis

2.4.1 Perception Module

For Perception Module, by training, the Object Detection is capable of correctly recognizing at least 30 types of objects, given current open-source dataset COCO [1] contains 80 categories and YOLO algorithm [2] can achieve high accuracy (68.9 mAP at [0.5]).

2.4.2 Planning Module

For Planning Module, current Large Language Models (e.g., ChatGPT, Doubao) are capable of reasoning about images for text generation and current VLA Models (e.g., OpenVLA [3]) are capable of predicting robot arm movements.

2.4.3 Action Module

For Action Module, 3D-printed and acrylic structures can withstand substantial forces without damage, and collision-free algorithms help prevent severe mechanical damage. Additionally, using silicone with a patterned surface increases the coefficient of friction, which enables the gripper to lift heavier objects with reduced risk of slipping.

2.4.4 Control Module

For the Control Module, the Raspberry Pi 4 should be=- capable of managing real-time communication and should be able to handle errors and failures, initiating recovery actions when necessary.

2.4.5 Remainings

The critical part is that it is unknown whether the pretrained Large Models on open source datasets can be adapted to our laboratory scene, which poses a risk.

3 Cost and Schedule

3.1 Cost Analysis

Cost Item	Unit Cost (USD)	Quantity	Total Cost (USD)
Hardware Components			
Raspberry Pi 4	$45.00	1	$45.00
PCB Printing	$20.00	3	$60.00
3D-Printed Gripper	$300.00	1	$300.00
Force Sensors (FSR402)	$10.00	2	$20.00
UR3e Robotic Arm	$0.00	1	$0.00
		Hardware Subtotal	$425.00
Labor			
Engineering Labor	$15.00/hr	200 hrs	$3,000.00
		Total Project Cost	**$3,425.00**

Table 1: Detailed Prototype Cost Breakdown

3.2 Schedule

Table 2: Project Timeline and Team Responsibilities

Week	Qi Long	Bingjun Guo	Yuxi Chen	Qingran Wu
Phase 1: Research & Investigation				
3/17	Literature Review: Guiding Long-Planning with VLM	Literature Review: Hierarchical Planning Foundation Model	Literature Review: Optimal force sensor placement for gripper design	Investigate common gripper design for robot arm

Continued on the next page

Table 2 – Continued from previous page

Week	Qi Long	Bingjun Guo	Yuxi Chen	Qingran Wu
3/24	Literature Review: VLA models, including OpenVLA	Literature Review: Improved VLA strategies, including ECoT	Investigate PCB design for sensor integration	Build the initial CAD model of the gripper
3/31	Set up server environments and experiment on OpenVLA model	Set up server environments and validate inverse kinematic method	Collaborate with Qingran sensor placement and gripper design	3D-print, assemble, test the gripper, and collaborate with Yuxi on sensor placement
Phase 2: Design				
4/7	**Team Collaboration: Design Document Composition**			
	Perception Module	Planning Module	Planning Module	Action Module

Continued on the next page

Table 2 – Continued from previous page

Week	Qi Long	Bingjun Guo	Yuxi Chen	Qingran Wu
Phase 3: Implementation				
4/14	Implement ECoT pipeline and experiment on ECoT-VLA model	Experiment the inverse kinematic method in lab environment	Create preliminary PCB layout	Improve the design and continue to assemble and test the gripper
4/21	Implement Object Locator and verify on testset	Start bonding the ECoT-VLA pipeline to the offline execution flow; feedback loop for user confirmation	Write ROS nodes for force sensors	Complete the assembly and test the functionality
4/28	Test Object Locator and ECoT-VLA model	Adapt and fine-tune the model regarding the specific tasks; force feedback loop	Execute end-to-end system testing on ROS and finalize PCB design	Figure out the connection between the gripper and the robot arm

Continued on the next page

Table 2 – Continued from previous page

Week	**Qi Long**	**Bingjun Guo**	**Yuxi Chen**	**Qingran Wu**
Phase 4: Testing & Integration				
5/5	**Integration Testing**			
	Software workflow testing with Bingjun	Software workflow testing with Qi	Hardware workflow testing with Qingran	Hardware workflow testing with Yuxi
5/12	**Full System Implementation**			
5/19	**System Testing and Debugging**			

4 Ethics and Safety

4.1 Ethical Issues

4.1.1 Development-Stage Ethical Concerns

The ACM Code of Ethics[4] emphasizes honesty about the capabilities and limitations of a system. To avoid misrepresenting the capabilities of our robot, we will specify its application scenarios, operational boundaries, and potential failing cases. In the meantime, since the planning stage of our robot is data-based, bias in training data that could result in unfair or unpredictable behavior is possible. Following IEEE and ACM guidelines, we will take fairness in consideration when determining our planning model (VLM) and test under diverse circumstances to mitigate bias. The IEEE Code of Ethics [5] also stresses that developers should accept responsibility for their technology's consequences, as the ACM Code emphasize on the robustness and usability of the system. To promote the sustainable development of our project, we will include logging and traceability features during the development to allow both developers and users to diagnose errors and attribute responsibility.

4.1.2 Misuse and Unintended Consequences

The project could be misused for unsafe or malicious tasks such as unauthorized modifications or weaponization. Responding to both the IEEE Code's concern [5] about physical abuse and ACM Code's[4] valuing on the public good, we will strictly restrict our robot's capability to conduct physical harm through e.g. limit the maximum operation speed or rejecting malicious language instructions. Another risk is that users might overestimate the robot's ability, leading to dangerous reliance on automation. We will provide clear user guidelines and training that ensure the users remaining awareness of the robot's limitations, responding to the ACM Code's requirement to foster public awareness of our technology.

4.2 Safety and Regulatory Standards

Safety is to be attached with primary significance during the process of both development and utilization. During the process, the robot arm could accidentally harm developers or users due to unexpected movements. In addition, bugs or adversarial commands caused by illusions of LLM could lead to unsafe robot actions. Below are several notable federal and industry safety standards to consider.

4.2.1 ISO 10218-1/2[6]

This standard mandates safety requirements for robotic arms, including emergency stop mechanisms, protective barriers, and collaborative safety features. The project will comply with these safety measures to ensure operational safety.

4.2.2 ISO/TS 15066[7]

As the project involves a robot interacting with human instructions, this standard provides guidelines on safe human-robot interaction, ensuring safe speeds, forces, and workspace conditions. Our kinetic and dynamic interpreter will comply to a preset safe constraint for motion output, and we will ensure that the instructions are given from a safe distance with respect to the robot's workspace.

4.2.3 ANSI/RIA R15.06[8]

This U.S. standard aligns with ISO 10218 and includes risk assessments, safety interlocks, and safe operation zones, all of which will be incorporated into the project's design.

4.2.4 Robot Manipulator General Safety Procedures[9]

Additional to the social standards, Illinois Robotics Group impose necessities to check the damage condition of robot arms and strict clothing regulations. Also, before the testing or instructing start, there should be loud and clear announcement that raise awareness and mental alert in the vicinity of everyone.

References

[1] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, *Microsoft coco: Common objects in context*, 2015. arXiv: 1405.0312 `[cs.CV]`. [Online]. Available: https://arxiv.org/abs/1405.0312.

[2] R. Khanam and M. Hussain, *Yolov11: An overview of the key architectural enhancements*, 2024. arXiv: 2410.17725 `[cs.CV]`. [Online]. Available: https://arxiv.org/abs/2410.17725.

[3] M. J. Kim, K. Pertsch, S. Karamcheti, *et al.*, *Openvla: An open-source vision-language-action model*, 2024. arXiv: 2406.09246 `[cs.RO]`. [Online]. Available: https://arxiv.org/abs/2406.09246.

[4] ACM. ""ACM Code of Ethics"." (2018), [Online]. Available: https://www.acm.org/code-of-ethics (visited on 03/14/2025).

[5] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 03/14/2025).

[6] ISO. ""Industrial robot safety bundle"." (2025), [Online]. Available: https://www.iso.org/publication/PUB200102.html.

[7] ISO. ""Robots and robotic devices — Collaborative robots"." (2016), [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:ts:15066:ed-1:v1:en.

[8] ISHN. ""ANSI/RIA R15.06-2012- The industrial robot safety standard"." (2018), [Online]. Available: https://www.ishn.com/articles/107815-ansiria-r1506-2012--the-industrial-robot-safety-standard.

[9] I. R. Group. ""Robot Manipulator Safety Rules"." (), [Online]. Available: https://robotics.illinois.edu/lab/robot-manipulator-safety-rules/ (visited on 03/14/2025).