Guessing Game with Robot Arm

Eldor Fozilov
Department of Computer Science and Engineering
eldorfozilov@unist.ac.kr

Gahyeon Shim
Graduate School of Artificial Intelligence
gahyeon@unist.ac.kr

Minji Kim
Graduate School of Artificial Intelligence
mzkim@unist.ac.kr

Seongjae Lee
Department of Electrical Engineering
lsjj1999@unist.ac.kr

Sunhong An
Department of Electrical Engineering
sunhong@unist.ac.kr

Youngbin Ki
Graduate School of Artificial Intelligence
youngbinki@unist.ac.kr

***Abstract*—This study introduces a robotic guessing game system that integrates multimodal perception and precise motion control. The system combines object detection, reasoning, and motion planning to infer and deliver target objects based on user-provided clues. Experimental evaluations validate its effectiveness in reasoning, object manipulation, and real-time operation, demonstrating its potential for dynamic human-robot interaction. The demonstration video and code is available at https://github.com/eldor-fozilov/guessing-game-with-robot**

I. Introduction

Robots are taking on diverse roles in daily life, such as Boston Dynamics' "Spot" handling inspections in hazardous areas and LG's Smart Home AI Agent managing home devices. These robots reduce human workload and enhance task efficiency by addressing specific operational needs.

Advancements in AI-based Human-Robot Interaction algorithms have increased expectations for robots to infer human intentions and respond accordingly. This technological progress promotes the development of human-centered interactions that enable robots to go beyond basic automation. The clues users provide play a key role in guiding the robot's reasoning and decision-making processes. To achieve this, robots are required to have multimodal reasoning capabilities that not only understand user-provided clues but also grasp environmental context.

Therefore, this project introduces a "Guessing Game" where a robot infers a user's intention using clues, feedback, and environmental context. The robot performs three main actions: (1) infers the target object from the user input, (2) validates the selection by the user feedback, and (3) delivers the object to the user.

The system leverages various mechanisms for perception and control, enabling seamless interaction between the robot and the user. The following section outlines the implementation and functionality of this interactive system.

II. Methodology

A. Overview

Our approach integrates multimodal perception and robot control, enabling a robot arm to understand user-provided clues and manipulate objects within its workspace. A guessing game user interface (UI) is implemented to facilitate intuitive interactions between the user and the robot. The UI is hosted

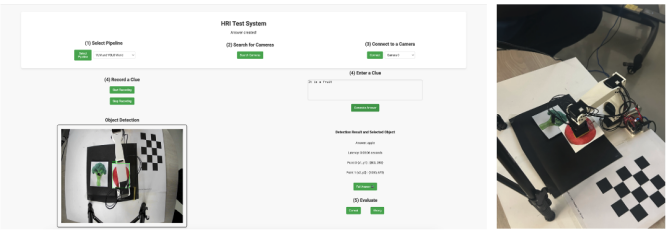

Fig. 1. User Interface for Guessing Game Interaction. The UI facilitates user input for clue-based interaction with the robot.

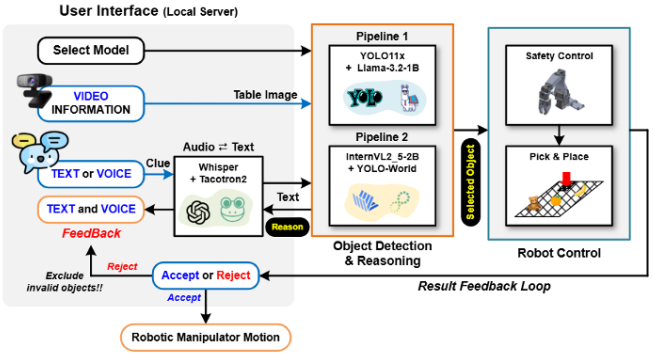

Fig. 2. System Workflow. The figure illustrates the overall process of the guessing game system.

on a local server using Flask, where user input is processed through Python functions triggered by a JavaScript file. This system, named the HRI TEST System, performs tasks using visual and textual clues provided by the user. The general workflow of our guessing game system is depicted in Fig. 2.

The system starts with the user placing objects on the table and providing a clue regarding the target object. For speech input, the clue is converted to text using the Whisper [1] model. The perception module processes an image of the table captured from a top-down view along with the user's clue. It outputs the detected objects, the selected object name, and its location, which is defined as the center of the object's bounding box. The control module is used to control the robot motion. First, the robot presents the inferred object to the user. Based on user feedback, if the robot delivers the correct object, it hands the object to the user. If the object is incorrect, it is placed in a designated area, and audio feedback explaining the robot's decision is generated using the Tacotron2 TTS [2] model. Incorrectly chosen objects are excluded from consideration in subsequent searches to prevent

repeated errors.

The perception module employs two distinct pipelines to handle object detection and reasoning, enabling the system to generalize to both predefined and novel objects. The control module enables the robot to pick up the inferred object using coordinate transformation and motion planning techniques. Detailed descriptions of these modules are provided in Sections II-B and II-C respectively.

B. Perception

To explore the ability of generalization to novel objects, we came up with two distinct approaches in the perception module: (1) **a closed-vocabulary approach**, which combines an object detection model with a predefined set of classes and a language model; (2) **an open-vocabulary approach**, which utilizes a general object detection model integrated with a vision-language model. The following subsections describe the details of each pipeline.

1) Pipeline 1 (closed-vocabulary approach): In this approach, the system uses YOLO11 [3] model, a robust and efficient object detection framework, and LLaMA-3.2-1B [4], a language model with strong reasoning capabilities. In this pipeline, the captured image is first processed by the YOLO11. The model identifies objects belonging to its predefined classes, providing their bounding box coordinates and labels. To avoid repeated errors, objects previously rejected by the user are filtered out during this stage.

The filtered object labels and the user-provided clue targeting a single object are then passed to LLaMA. The language model analyzes the clue in context, selects the object most relevant to the clue, and provides a brief explanation for the selection. The explanation serves as feedback to the user. The following prompt is used to guide the language model's output:

> You are an assistant whose task is to identify the correct object from this list {detected_objects} that aligns best with the following clue: '{clue}'.
> Take a moment to think about each object's properties, considering how the clue relates to them, directly or indirectly.
> Provide your response in the following format:
> Selected Object: <name of the correct object>
> Explanation: <brief explanation of how the object's properties align with the clue>
> The response should be structured exactly like this, with "Selected Object:" followed by the object's name, and "Explanation:" followed by a brief explanation.
> Response:

The output of this pipeline includes the inferred object name and an explanation, both of which are passed to subsequent system stages.

2) Pipeline 2 (open-vocabulary approach): Pipeline 2 integrates InternVL-2.5-2B [5] model, which excels at multimodal reasoning tasks, and YOLO World, an efficient open-vocabulary object detection framework. The captured image and the user-provided clue are input to the InternVL-2.5.2B, which first detects and classifies all objects in the image. It then selects the most appropriate one that matches the clue and returns a brief explanation for its selection. We use the following prompt to accomplish this task:

> You are an assistant whose task is to analyze the given image and the following clue: '{clue}'.
> 1. Identify and list all the objects present in the image.
> 2. From the list of identified objects, select the one that best aligns with the clue, while ignoring these excluded objects if present: {excluded_objects}.
> 3. Provide a brief explanation of why the selected object matches the clue.
> Format your response as follows:
> All Identified Objects: <comma-separated list of objects>
> Selected Object: <name of the correct object>
> Explanation: <brief explanation>
> Response:

The model outputs a list of identified objects, the selected object, and an explanation for the decision. The inferred object name and the captured table image are passed to the YOLO World. YOLO World provides the bounding box coordinates of the selected object, which are used in the robot's subsequent actions.

C. Control

1) Coordinate Transformation: In the Guessing Game, the robot operates in a coordinate system where the origin is located at the center point of the robot base. However, the object location from the perception module is defined in a 2D image coordinate system. Therefore, a transformation from the 2D coordinates to the 3D coordinates in the robot's coordinate system is necessary.

The intrinsic matrix, M_{int}, is fixed during the camera manufacturing process. It is calculated once using a ChArUco board. In contrast, the extrinsic matrix, M_{ext}, which represents the transformation between the camera and the world T_{cb}, should be recalculated whenever the camera position changes. At the start of each game, the camera captures the current view to compute T_{cb} by detecting a checkerboard pattern using the *OpenCV* library [7].

Since the calibration board's position and orientation relative to the robot origin T_{rb} are known, the 2D image point P_{img} can be transformed into a 3D world point in the robot coordinate system P_{robot}. Equation (1) shows the process of coordinate transformation. As depth information is unavailable with an RGB camera, the fixed height of the camera from the workspace Z_{cam} is used, as images are consistently captured from a top-down perspective.

$$P_{robot} = T_{rb} \cdot T_{cb}^{-1} \cdot M_{int}^{-1} \cdot P_{img} \cdot Z_{cam} \tag{1}$$

2) Robot Control: To enable the robot end effector to move to the target position, an Inverse Kinematic (IK) solver is applied, calculating joint values for the 6-DoF robot arm. We utilize the *IKPy* library [8], a lightweight and fast open-source solution. A trajectory is generated through linear interpolation between the current position and the target position, with evenly spaced steps. This trajectory is used to control the positions of the Dynamixel motors.

To address inaccuracy in reaching the target position, we apply PID (Proportional-Integral-Derivative) control. The error is defined as the difference between the target and current PWM values for each joint. This error is used to calculate control inputs for the next step that minimizes positional error. Once the positional error falls below a defined threshold, the

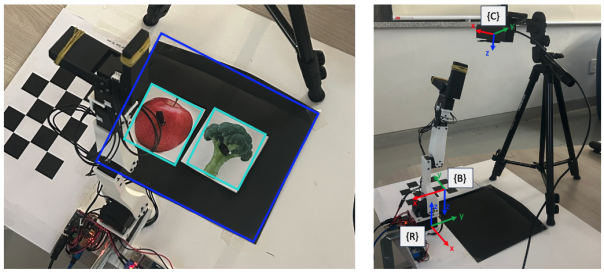

Fig. 3. Experiment Setup. (Left) The blue box represents the predefined workspace for object placement. The cyan box shows 2D-printed images with attached pillars for grasping. (Right) The experimental setup with coordinate definitions for the robot (R), checkerboard (B), and camera (C).

Fig. 4. Object Classes in Scenarios. The figure shows 12 object classes used in the experimental scenarios, including both YOLO and non-YOLO objects. Each class is represented by a corresponding 2D-printed image.

robot completes the current action and moves on to the next action. In the experiment, the PID constants are set to $K_p = 0.5$, $K_i = 0.1$, and $K_d = 0.01$. The PID formula is shown in Equation (2):

$$u(t) = K_p \cdot e(t) + K_i \cdot \int e(t)\,dt + K_d \cdot \frac{de(t)}{dt} \tag{2}$$

The $e(t)$ represents the positional error at time t, and $u(t)$ is the control signal output used to adjust the motor positions

Additionally, to prevent the robot from colliding with other objects or unintended areas, the orientation of the end effector is fixed downward during the grasping process. It ensures stable and accurate pick-and-place action, avoiding collision with surrounding objects.

III. Evaluation

To evaluate the performance of our system, we conduct a series of experiments designed to measure the performance of both the perception and the control module.

A. Experiment Design and Setup

1) Assumption: The experimental assumptions are illustrated in Fig. 3, with details outlined below:

Object Design: Since the provided gripper is not suitable for handling real objects, we use 2D-printed images. These images are printed large enough for the perception module to detect and a 3D-printed pillar is attached to it, enabling the robot to grasp them (cyan box in Fig. 3).

Object Placement: All objects are arranged within a predefined workspace (blue box in Fig. 3), considering both the operating range of the robot arm and the visible range of the camera.

Selectable Objects: All objects should be included in the class set used to train the YOLO. However, to evaluate the open-vocabulary object detection ability of YOLO-World, we use additional objects not included in this class set.

2) Experiment Settings: The experimental setup and environment for the Guessing Game are detailed as follows:

Environment : The experimental setup, illustrated in Figure 3, demonstrates the configuration for conducting the Guessing Game using the robot arm and camera. The figure visually displays the origins and coordinate axes of the robot (R), checkerboard (B), and camera (C). We use an FHD webcam and a position-controlled Koch v1.1, a low-cost robot arm, with a modified gripper. The original gripper is a pointed design, which makes unstable grasping. We modify the gripper to increase the contact area, resulting in more stable grasping. Additionally, rubber bands are wrapped around the gripper tips to enhance friction and further improve grip stability.

Scenarios : A total of 20 experimental scenarios are designed. These scenarios include 12 predefined object classes (shown in Fig. 4), encompassing both classes used to train the YOLO and those not used when training YOLO. Each scenario involves a set of three objects. To evaluate system performance, two levels of clues are generated for each scenario using GPT-4o, with varying difficulties based on the object set.

B. Perception

1) Evaluation Metrics: The experiments on the perception module aim to: (1) measure the classification accuracy and generalization ability of Pipeline 2, and (2) compare the reasoning performance of both pipelines.

For the classification accuracy, the metric is defined as:

$$Acc = \frac{N_{classify}}{N_{total}} \tag{3}$$

The $N_{classify}$ is the number of successful classifications, and N_{total} is the total number of classification attempts. The criteria for successful classification differs for each pipeline. For YOLO, all objects in the image must be correctly classified. For YOLO-World, the class label of the object should match the class provided by the vision-language model.

For the reasoning, we evaluate the model's ability to correctly infer the target object based on user-provided clues. Two metrics are used: Acc_{c1}, which measures the success rate using a single challenging clue, and Acc_{c2}, which measures the success rate when an additional easier clue is provided after an incorrect inference with the initial clue. These metrics are defined as follows:

$$Acc_{c1} = \frac{N_{c1}}{N_{total\_c1}}, \quad Acc_{c2} = \frac{N_{c2}}{N_{total\_c2}} \tag{4}$$

The N_{c1} and N_{c2} are the number of successes for each clue type, and N_{total_c1} and N_{total_c2} are the total attempts.

TABLE I
EXPERIMENT RESULTS ON PERCEPTION MODULE

Pipelines	Detection	Reasoning	
	Acc. (%)	Acc_{c1}. (%)	Acc_{c2}. (%)
YOLO + LLM (1)	80.0	75.0	20.0
VLM + YOLO-World (2)	100.0	75.0	100.0

2) Results: The experiment results are summarized in Table. I. For classification, Pipeline 1 achieves an 80% accuracy rate, whereas Pipeline 2 successfully classifies objects in all test runs. YOLO struggles with objects outside its predefined class set, while YOLO-World successfully classifies those objects. This indicates that YOLO-World has broader classification capabilities, enabling it to handle a wider variety of objects.

For reasoning, a target object is randomly selected, and the same target object is given for both pipelines to ensure a fair comparison. A challenging clue related to the target object is provided via the UI. If the model incorrectly infers the target object, an easier clue is given as additional feedback. Both pipelines achieve 75% on Acc_{c1}. For Acc_{c2}, Pipeline 1 achieves 20% accuracy, while Pipeline 2 achieves 100%. This significant difference is due to YOLO-World's ability to detect novel objects not included in the predefined class set since Pipeline 1 fails when the target object does not belong to the predefined class set.

C. Robot Motion Control

1) Evaluation Metrics: The control module is evaluated through tests in which the robot is required to pick up a target object and place it at a predefined location. The goals of the experiments are: (1) to demonstrate the integration of the control module with the perception module, and (2) to measure the accuracy and validity of the control module. First, we evaluate the integrated perception-and-control system by measuring its task success rate. Next, we perform an additional experiment focused solely on the control system to validate its functionality and accuracy. The evaluation metrics are defined as follows:

Task Success Rate: The task success rate is defined as the average percentage of successful task completions over the n test runs. A test is considered successful if the robot picks up the target object and places it at the designated location.

Position error: The position error is defined as the average Euclidean distance between the target point and the actual point where the robot moved.

2) Results: The fully integrated system including perception and control is tested to validate the integration of the control module with the perception module. For each pipeline, three objects are placed in the workspace, following the scenarios used in perception module evaluation. A test is considered successful if the robot reasons the right object among three objects based on the clue and does the pick-and-place action. A total of n = 20 runs are conducted for each pipeline.

TABLE II
EXPERIMENTS ON FULL INTEGRATED SYSTEM AND CONTROL MODULE

Method	Success Rate (%) (↑)	Position Error (mm) (↓)
Pipeline 1 + Control	75.0	-
Pipeline 2 + Control	75.0	-
Control only	86.7	0.016

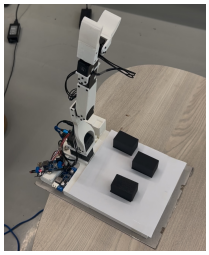

Fig. 5. Setup for Extra Experiment. The setup includes predefined target locations and controlled conditions to evaluate the robot's motion accuracy.

As shown in the first two rows of Table II, both Pipeline 1 and Pipeline 2 achieve a task success rate of 75%, successfully completing 16 out of 20 runs. Each run is completed within one minute, demonstrating the system's capability to operate in real time. It is observed that for Pipeline 2 integrated with the control system, the majority of failure cases result from robot malfunctions occurring as the experiments progress. Despite Pipeline 2 demonstrating superior reasoning performance with novel objects, the robot frequently fails to reach the target point and exhibits undesirable poses due to motor malfunctions.

We conduct additional experiments using a robot from another group to further evaluate the validity of our control module. The experimental setup is shown in Fig. 5. Three objects are placed within the workspace, and a run is considered successful if the robot picks the target object and places it at the designated location. A total of n = 60 runs are performed.

The first row of Table II presents the task success rate and position error results. The average position error across all experiments is within 2 cm, demonstrating that our control method which leverages PID control and the IK solver provided by the *IKPy* library, is accurate. For task completion, the robot successfully picks the target object in every run. However, there are cases where the robot fails to place the object at the designated location.

IV. CONCLUSION

We develop a Human-Robot Interaction system that effectively combines multimodal perception and control for intuitive interaction. Results show robust performance in object recognition, reasoning, and manipulation, highlighting its applicability to real-world tasks. The system's ability to interpret user clues and respond to user feedback underlines its potential for applications in personalized assistance, smart homes, and collaborative workspaces. Future work can focus on expanding its capabilities for complex environments with mobile robots and real-world objects.

REFERENCES

[1] OpenAI, "Whisper: OpenAI's Automatic Speech Recognition System," GitHub repository, 2023. [Online]. Available: https://github.com/openai/whisper

[2] Coqui, "TTS: A deep learning toolkit for Text-to-Speech," GitHub repository, 2023. [Online]. Available: https://github.com/coqui-ai/TTS

[3] R. Khanam and M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," arXiv preprint arXiv:2410.17725, 2024. [Online]. Available: https://arxiv.org/abs/2410.17725

[4] Meta AI, "LLaMA 3.2-1B: A Multilingual Large Language Model," GitHub repository, 2024. [Online]. Available: https://github.com/meta-llama/llama3

[5] OpenGVLab, "InternVL2_5-2B: A Multimodal Large Language Model," GitHub repository, 2024. [Online]. Available: https://github.com/OpenGVLab/InternVL

[6] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan, "YOLO-World: Real-Time Open-Vocabulary Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 1234-1243. [Online]. Available: https://arxiv.org/abs/2401.17270

[7] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[8] P. Manceron, "IKPy," GitHub repository, [Online]. Available: https://github.com/Phylliade/ikpy, DOI: 10.5281/zenodo.6551105, License: GPL-2.0.