Milestone report for Casabot: The Fusion of AI and Robotics

Group 4: 12111028 Wang Junyang 12111026 Zhou Jingdong 12110510 Deng Haowen 11910412 Mao Xinke 12110501 Ji Yibing 12111127 Daniel Tan SioaHen 12111128 Ng Wooi Cheng

***Abstract*—This project aims to develop a new type of collaborative robot arm that enhances home automation by understanding and executing tasks through natural language interaction. At the same time, it also needs to train a machine learning model to recognize different types of items and sort them to the correct location. Additionally, we will access ChatGPT to achieve various types of command conversion, to achieve key input, voice input and even gesture recognition input, in order to achieve the convenience of this robot.This project will combine computer vision, deep learning systems, robotic arms, and the basic principles of robotic arms to achieve. By training a robotic arm to sort items, the efficiency and accuracy of daily household work can be greatly improved.**

I. Introduction

For some people with disabilities, daily work at home can be very inconvenient, so an intelligent assistant that can enter the home is very necessary. This project is dedicated to designing a desktop-level robot arm based on visual grasping, using machine learning to train the model, and finally connecting chatgpt to realize various forms of instruction input to complete the purpose of grasping. We plan to complete this project through the following steps:

Collecting visual grasping datasets: We need to collect item grasping datasets of different types to train the machine learning model.

Data preprocessing: For the collected image data, we need to do some preprocessing, such as image enhancement and image cropping.

Training the model: Train a machine learning model using a deep learning framework (such as TensorFlow or PyTorch) to recognize different types of items.

Deploying the model: Deploy the trained model to the robotic arm and write control programs so that it can sort items according to the recognition results.

Connecting to ChatGPT: Access ChatGPT to realize command conversion and input, such as summarizing the synonyms that may be different in the population, and realizing key input, voice input and even gesture recognition input.

II. Problem Statement

A. Object identification

The following types of items need to be judged differently, and the corresponding trajectory planning process should be made.

Items that need to be sorted out: For the desktop-level intelligent robot arm designed by us, the first thing to be identified is the objects that need to be sorted out on the desktop, such as pens, books, and so on; Some special items also require special grasping strategies. Also take into account some dynamic factors, such as some items in use when moving in the hand, also need to be identified

Obstacles to avoid: When working on the desktop, consider that there are some items that do not need to be appropriated, such as lamps, or personal computers, etc., these items can not be crashed, so it is necessary to accurately identify these objects; Additionally, places that need to be used to place items, such as bookshelves, pen holders, etc., also need to be taken into account, and to design a grasping placement strategy that adapts to different environments.

B. Implement ChatGPT

To implement ChatGPT as a medium for instruction conversion, allowing it to receive commands and translate them into instructions executable by our robotic arm.

Define Instruction Format: Determine the format of instructions for ChatGPT to understand and process. Instructions should include action to be executed (e.g., grasp, release, move) and descriptions of target objects or positions.

Develop Instruction Parser: Write a program or script to parse commands received by ChatGPT and convert them into instructions understandable and executable by the robotic arm. This may involve natural language processing (NLP) techniques and logic programming.

Robotic Arm Control Interface: Determine the control interface and communication protocol for the robotic arm, enabling commands to be sent to the arm from a computer or embedded system, and retrieving execution results.

Instruction Executor: Write a program or script to translate instructions parsed by ChatGPT into commands recognizable by the robotic arm control interface, and send them to the arm for execution. This may involve using programming languages and control libraries.

Integration Testing: Conduct integration testing of the entire system to ensure that ChatGPT correctly parses instructions and instructs the robotic arm to perform the corresponding actions. Debugging and optimization may be required during testing.

Real-time Interaction: Deploy the system in a real-world environment and implement real-time interaction functionality, allowing users to control the robotic arm by conversing with ChatGPT.

III. Literature Review

A. *Machine Learning for Visual Grasp*

The challenge of visual grasping in cluttered environments is central to advancing robotic manipulation, especially in home settings where a robot must deal with a variety of objects. **PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes** [5] directly addresses this challenge by introducing a robust framework for pose estimation, which is critical for precise object manipulation. This method's ability to differentiate and accurately estimate the position and orientation of multiple objects in real-time makes it highly relevant for our project's goal of enhancing a robot's grasping capabilities (Xiang et al., 2017).

Additionally, the integration of deep learning for object recognition within robotic systems is explored in **Integration of Deep Learning-Based Object Recognition and Robot Manipulator for Grasping Objects** [3]. This paper demonstrates effective strategies for combining CNN-based vision systems with robotic arms to achieve reliable grasping actions. The approaches discussed here will inform our development of a machine learning model capable of supporting sophisticated manipulation tasks (Integration of Deep Learning-based Object Recognition and Robot Manipulator for Grasping Objects, 2019).

Furthermore, the comprehensive review provided in **Vision-Based Robotic Grasping from Object Localization, Object Pose Estimation to Grasp Estimation for Parallel Grippers** [1] offers a detailed overview of the progression from basic object recognition to advanced grasp planning. This review is instrumental for understanding the state-of-the-art techniques and existing challenges in robotic grasping, ensuring our project is built on a solid foundation of current knowledge (Du et al., 2020).

B. *Integrating ChatGPT for Command Input*

The potential for natural language processing to enhance robot control is being showed in **ChatGPT Empowered Long-Step Robot Control in Various Environments: A Case Application** [4]. This paper showcases the practical application of ChatGPT in controlling robotic actions over extended sequences, which is particularly pertinent to our project as it seeks to enable complex task execution through simplified command inputs (ChatGPT Empowered Long-Step Robot Control in Various Environments: A Case Application, 2023).

Moreover, **ROSGPT: Next-Generation Human-Robot Interaction with ChatGPT and ROS** [2] reveals advancements in integrating large language models with robotic operating systems for improved human-robot interactions. The integration techniques discussed will be crucial for our project, as they provide a blueprint for embedding ChatGPT into our robotic system, allowing for more intuitive and versatile user commands (Koubâa, 2023).

IV. Technical Approach

Primarily, the tasks our team will undertake involve object recognition and ChatGPT implementation. Object recognition involves the identification and classification of various objects within images or video streams, a fundamental task with applications spanning from industrial automation to augmented reality. Concurrently, integrating ChatGPT, a cutting-edge language model, presents opportunities to enhance user interaction and automate tasks through natural language processing. We aim to explore the synergistic implementation of both Object Recognition and ChatGPT. By leveraging the capabilities of object recognition algorithms alongside the conversational prowess of ChatGPT, we endeavour to develop a versatile system capable of understanding and responding to the world in both visual and textual modalities.

A. *Object Identification.*

Object recognition, a pivotal task in computer vision, aims to identify and classify objects within images or video streams. In collaborative robotics applications where humans work alongside robots, object recognition plays a crucial role in ensuring safety. Robot arms equipped with vision systems can detect and avoid collisions with humans or other objects in the workspace, minimizing the risk of accidents and injuries. In warehouse and logistics operations, object recognition enables robot arms to sort and pack items accurately and efficiently. Similarly, in house tidying, with the help of object recognition, smart tidying systems are able to automatically identify and categorize various items within the home, sorting them into designated storage areas or containers efficiently.

Among the myriad of algorithms developed for this purpose, YOLO (You Only Look Once) stands out as a pioneering approach renowned for its speed and accuracy. Unlike traditional object detection methods that rely on sliding windows or region proposal networks, YOLO approaches object detection as a single regression problem, directly predicting bounding boxes and class probabilities for each object within an image. The key principle behind YOLO is its unified architecture, which divides the input image into a grid and predicts bounding boxes and class probabilities directly from the grid cells. By incorporating global context information into the prediction process, YOLO is able to detect objects with high accuracy and efficiency, making it suitable for real-time applications such as autonomous driving, surveillance, and robotics. Furthermore, YOLO is capable of detecting multiple objects within a single image, providing comprehensive scene understanding in a single pass.

YOLO is, in fact, one of the most potent methods for object recognition. After several iteration and optimization of the function, there are now different updated version of YOLO (YOLOv1 to YOLOv8). YOLOv8 will have a high possibility to be mainly utilized in this project as this newest version possesses the most updated and optimized function of object recognition.

B. ChatGPT Implementation.

Implementing ChatGPT as a medium for instruction conversion enables seamless communication between users and robotic arms for executing various tasks and commands. Integrate the ChatGPT model into our system's architecture enable users more easily to interact with the system, in another word, more user friendly, while ensuring the accuracy of executing commands.

There are several crucial capabilities of ChatGPT implementation. Firstly, text generation is one of the most basic function of ChatGPT. OpenAI's text generation models, such as GPT-4 and GPT-3.5, are trained to comprehend both natural and formal language. These models, including GPT-4, produce text outputs in response to inputs, often referred to as "prompts". Crafting a prompt essentially acts as programming for a model like GPT-4, typically involving instructions or examples to guide task completion. These models find utility across diverse tasks, including content and code generation, summarization, conversation, creative writing, and more.

OpenAI's text generation models undergo pre-training using extensive text data. To optimize their usage, we incorporate instructions and occasionally multiple examples within a prompt. Employing demonstrations to illustrate task execution is commonly known as "few-shot learning." Fine-tuning enhances few-shot learning by training on a significantly larger number of examples than what can be accommodated within the prompt, thereby enabling superior performance across various tasks. Following fine-tuning, fewer examples are required in the prompt, leading to cost savings and facilitating faster response times.

Other than that, text embeddings measure the relatedness of text strings. An embedding consists of a list of floating-point numbers, forming a vector. The proximity between two vectors indicates their correlation, with short distances implying strong correlation and long distances indicating weak correlation.

Besides, the function calling enables particular tasks or actions to be carried out upon receiving commands. This capability enables it to understand commands and invoke the suitable routines for tasks such as object grasping.

Last but not least, text-to-speech and speech-to-text conversion are able to enhance the interaction between users and robot. For the text-to-speech, the Audio API integrates a speech endpoint leveraging the TTS model with the functionalities include converting written blog posts into spoken narratives, generating spoken content in multiple languages, and providing real-time audio output through streaming. For the speech-to-text, the Audio API provides two speech to text endpoints, transcriptions and translations.

V. Results

To evaluate the effectiveness of our proposed system, we will employ a comprehensive approach that encompasses various aspects and datasets:

A. Performance Metrics

We will measure the accuracy, speed, and efficiency of the robotic arm in executing tasks within the simulation. Accuracy will be determined by the percentage of correctly identified and placed items. Speed will be evaluated based on the time taken to complete tasks, including object recognition and grasping. Efficiency will be assessed by comparing the performance of the robotic arm with manual sorting methods.

B. Simulation Dataset Evaluation

We will evaluate the quality and coverage of the simulation dataset to ensure it adequately represents various types of items and scenarios. Techniques such as cross-validation will be utilized to verify the robustness and generalization ability of the trained models.

C. Real-time Simulation Testing

We will simulate real-world daily tasks within the simulation environment and observe the performance of the robotic arm. We will test its adaptability to different scenarios and environments and its ability to handle obstacles.

D. Performance Comparison

Finally, we will compare our system with baseline methods to evaluate its advantages and limitations in the simulated environment. Through these evaluation methods, we aim to gain comprehensive insights into the performance of our proposed system in a simulated environment and guide future improvements

References

[1] Guoguang Du, Kai Wang, Shiguo Lian, and Kaiyong Zhao. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *Artificial Intelligence Review*, 54(3): 1677–1734, 2021.

[2] Anis Koubaa. Rosgpt: Next-generation human-robot interaction with chatgpt and ros. 2023.

[3] Hyunsoo Shin, Hyunho Hwang, Hyunseok Yoon, and Sungon Lee. Integration of deep learning-based object recognition and robot manipulator for grasping objects. In *2019 16th international conference on ubiquitous robots (UR)*, pages 174–178. IEEE, 2019.

[4] Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Chatgpt empowered long-step robot control in various environments: A case application. *IEEE Access*, 2023.

[5] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.