ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

Visual Chatting and Real-Time Acting Robot

Team #37

HAOZHE CHI
(haozhe4@illinois.edu)
MINGHUA YANG
(minghua3@illinois.edu)
JIATONG LI
(jl180@illinois.edu)
ZONGHAI JING
(zonghai2@illinois.edu)

TA: Enxin Song

April 19, 2024

1 Introduction

1.1 Problem Statement

Blind individuals often face significant difficulties when navigating unfamiliar environments, such as finding water dispensers in large public spaces. Additionally, there is a risk of injury from interacting with devices that dispense hot water. The emergence of large language models (LLMs) and large visual language models (LVLMs) offers a promising avenue for developing innovative solutions to these challenges.

1.2 Solution Overview

We propose to create an AI-enhanced robotic service system designed to assist blind people by guiding them to water dispensers, providing real-time vocal instructions for navigation and safety, and autonomously refilling water bottles. This system will integrate a camera mounted on the user's head for visual input and use speech-to-text AI technology to interpret verbal commands. The core of our solution is the BLIP-2 visual language AI model, which will process both visual and textual inputs to generate actionable guidance. Additionally, a text-to-speech AI will transform text outputs into auditory instructions, thereby facilitating the user's interaction with their environment. Our system comprises the following key components:

- **Real-Time Visual and Verbal Input Processing:** A combination of a head-mounted camera and speech-to-text AI captures and analyzes the user's surroundings and voice commands.
- **Dynamic Guidance and Interaction:** The BLIP-2 model will provide navigation assistance, warn of potential dangers, and instruct on interacting with a water dispenser.
- **Autonomous Assistance:** A Universal Robot Arm UR3e, controlled by the Robot Operating System and instructed by the Raspberry Pi Auxiliary System, will autonomously refill the user's water bottle.
- **User Communication:** Audio feedback and instructions will be delivered through a Bluetooth headset, ensuring clear and effective communication.

Operational Process When a blind individual approaches a water dispenser, the system triggers a specific sequence of actions:

1. The Vision Language AI model guides the user to the water dispenser.
2. The Raspberry Pi Auxiliary System will then provide audio instructions to help user place their bottle in a designated location.
3. Subsequently, a robot arm, following instructions from the Raspberry Pi system, securely grasps the bottle, fills it with water from the dispenser, and then returns the filled bottle to the user.

Challenges and Innovation Integrating LVLMs within a robotic framework presents unique technical challenges, especially in accurately translating AI-generated instructions into precise robotic movements. This project not only aims to address a tangible issue but also seeks to advance the field of AI and robotics by exploring new applications of visual language models in assistive technology.

1.3 Visual Aid

The visual illustration of our AI-enhanced robotic service system is shown in Figure 1.

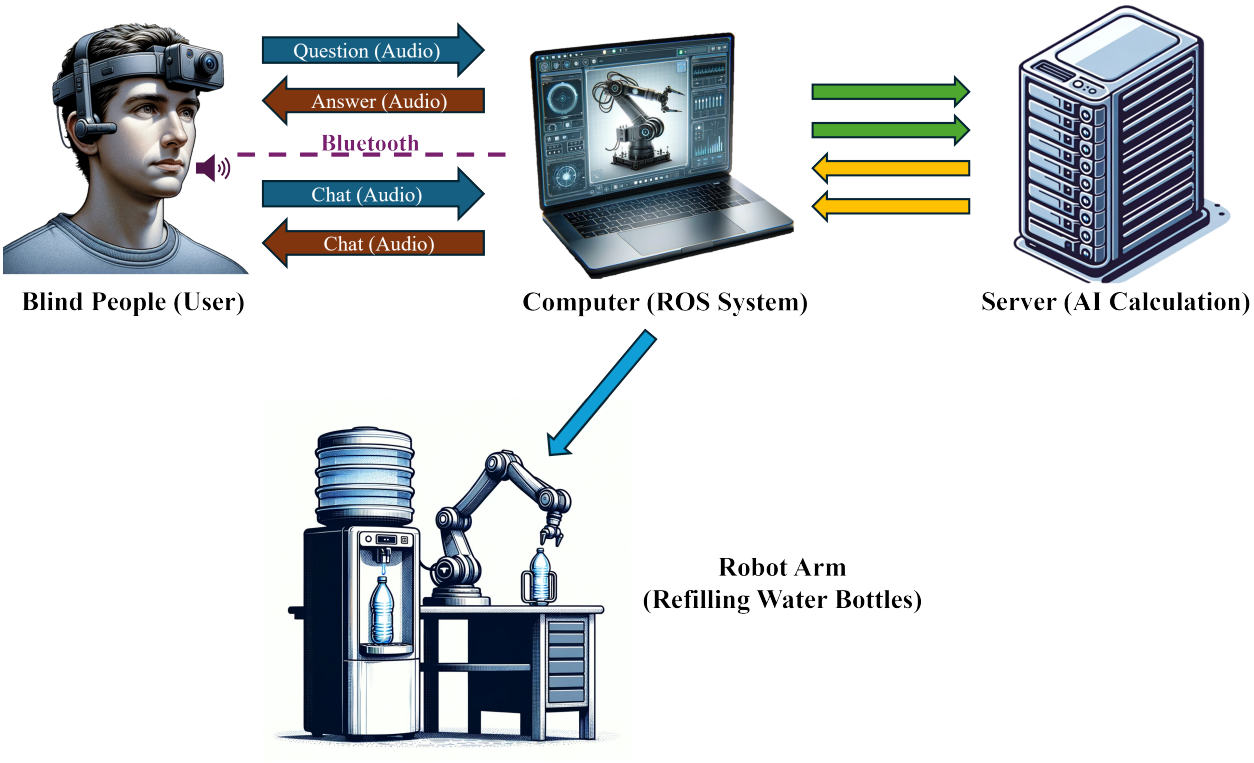

Figure 1: Visual Illustration of the AI-enhanced Robotic Service System

1.4 High-level Requirements List

1. **Server Specifications:** A high-performance server equipped with a GPU boasting a minimum of 24 GB of memory is essential for efficiently running the Visual Language AI model.
2. **Robotic Arm:** The project demands a robotic arm with a minimum of six joints for versatile movement and object manipulation capabilities. An example of such hardware is the Universal Robot UR3e.

3. **Camera System:** We require four cameras capable of capturing real-time images. The primary camera will provide visual feedback from the perspective of blind users, while the other three cameras will be used to detect the presence and position of water bottles.

2 Design

2.1 Block Diagram

The overall block diagram of our AI-enhanced robotic service system is shown in Figure 2.

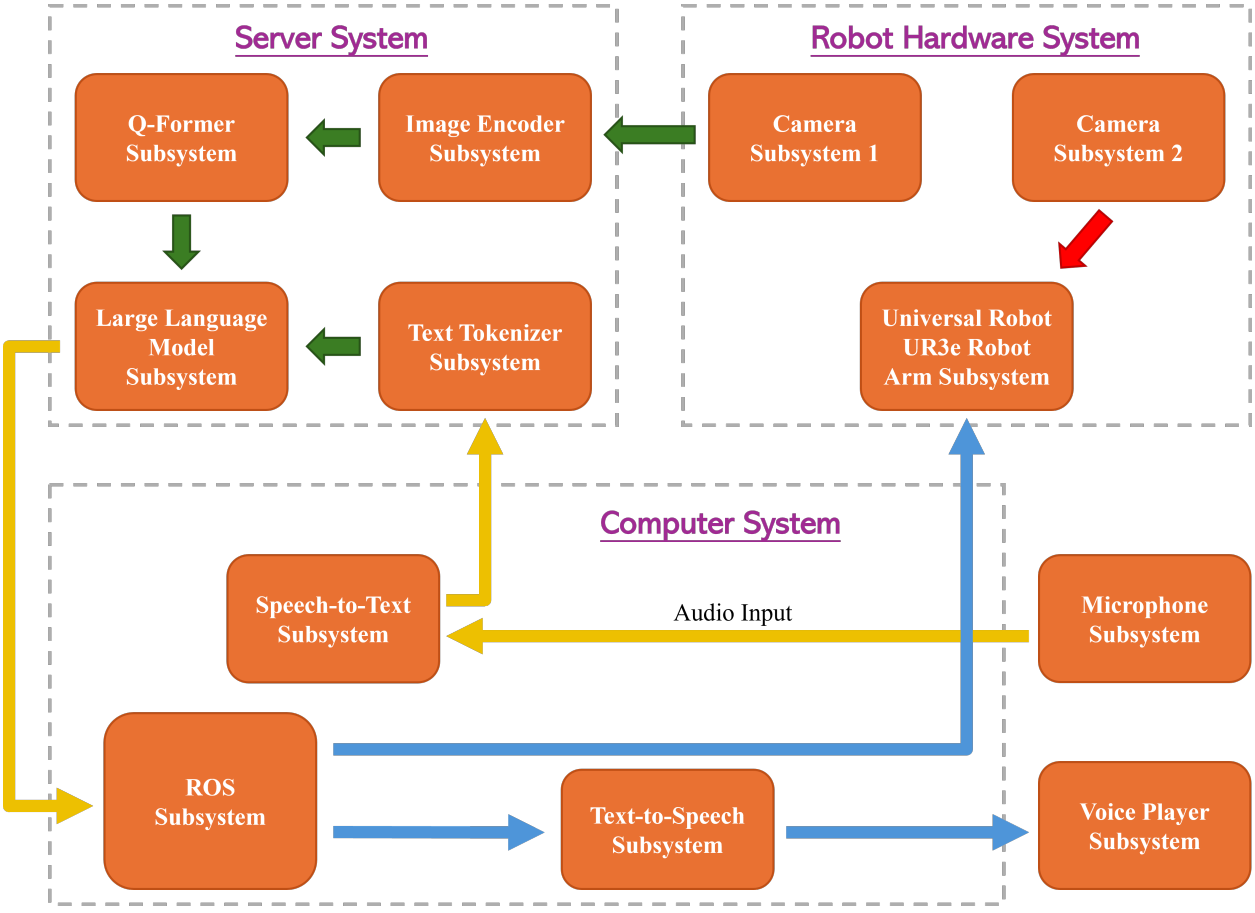

Figure 2: Block diagram of the AI-enhanced robotic service system (Green arrow: visual flow; Yellow arrow: text flow; Red arrow: attach itself to; Blue arrow: instruction flow), in which Camera Subsystem 1 is for blind people and Camera Subsystem 2 is for robot arm.

2.2 Subsystems

2.2.1 Camera Subsystem

The Camera Subsystem is a pivotal element in our robotic framework, acting as the primary data collection point. Utilizing an iPhone camera mounted on the user's head, this subsystem captures the user's environment and streams video in real-time to a connected Mac. This setup ensures a continuous flow of high-resolution visual data to the Image Encoder Subsystem. The system's contribution is quantified by its ability to deliver high-resolution video under varying lighting conditions and maintain a seamless frame rate essential for subsequent processing stages. The interface with the Image Encoder Subsystem is defined by the video resolution, frame rate, and the real-time data transfer rate necessary for effective processing.

Requirements	**Verification**
1. Stream video in real-time with minimal delay to ensure interactive processing.	A. Measure the time delay from capture on the iPhone to receipt on the Mac to ensure it's within acceptable limits for real-time interaction. B. Test the system during various interactive scenarios to validate consistent performance.
2. Maintain high image resolution suitable for detailed visual processing.	A. Verify the resolution of captured images meets the predefined standard. B. Test image quality under various lighting conditions.
3. Ensure reliable data transfer to the Image Encoder subsystem with minimal latency.	A. Measure the latency between image capture and data reception by the Image Encoder. B. Conduct stress tests to ensure data integrity and transfer rate under peak loads.

Table 1: Camera Subsystem Requirements and Verification

2.2.2 Image Encoder Subsystem

The Image Encoder Subsystem is an integral part of the robotics framework, responsible for converting visual input into a format suitable for advanced analysis. Leveraging the Vision Transformer (ViT) structure, specifically a **ViT-L/14** model, this subsystem provides a streamlined and feature-enriched representation of images captured by the Camera subsystem. It processes images through 32 queries of 768 dimensions each, aligning with the Q-Former's specifications for efficient interfacing. The subsystem's output, a

compressed 32×768 matrix denoted as Z, presents a more efficient alternative to the initial 257×1024 ViT-L/14 image features. By reducing data dimensionality, it plays a crucial role in optimizing the computational workflow and facilitating swift data exchange with the Q-Former subsystem.

Requirements	**Verification**
1. The ViT-L/14 must compress images into a 32×768 representation, matching the Q-Former's hidden dimension.	A. Verify the output dimensionality post-encoding against the 32×768 requirement. B. Ensure consistency of the output dimensions across multiple image inputs.
2. The subsystem must process images from the Camera subsystem without exceeding a 200ms encoding latency.	A. Measure the encoding time from image capture to feature output. B. Confirm latency is under 200ms for a range of image complexities.
3. The Image Encoder should maintain an encoding accuracy of 90% when evaluated against a standard set of image features.	A. Compare encoded features with a benchmark feature set. B. Calculate accuracy percentage and confirm it meets the 90% threshold.

Table 2: Image Encoder Subsystem Requirements and Verification

2.2.3 Q-Former Subsystem

The Q-Former Subsystem serves as a crucial component in the processing pipeline of our design, utilizing transformer architecture to elevate visual data into abstract representations. This subsystem ingeniously employs attention mechanisms to enhance the visual features received from the Image Encoder subsystem before passing them on to the Large Language Model (LLM) subsystem. Quantitatively, it boasts two transformer submodules that share self-attention layers to refine features from varying image resolutions. The image transformer submodule is tasked with the visual aspect, and the text transformer handles the encoding and decoding of textual information. Through this setup, the Q-Former ensures that the interaction between visual and textual data is not only seamless but also optimized for the highest efficiency in real-time processing.

Requirements	Verification
1. The subsystem must integrate visual features from the Image Encoder with a maximum latency of 100ms.	A. Measure the time taken from receiving the input to producing the output using a high-precision timer. B. Verify latency consistency across multiple test runs.
2. The Q-Former should handle image resolutions up to 4K UHD (3840x2160 pixels) without degradation of attention mechanism performance.	A. Test the subsystem with images of increasing resolutions, up to 4K UHD. B. Assess if the attention mechanisms operate within specified performance parameters.
3. The subsystem must accurately process visual and textual data, demonstrating a minimum accuracy of 90% in feature enhancement relevance.	A. Conduct a series of feature extraction and refinement tests. B. Utilize a predefined validation set to evaluate the accuracy of the enhanced features.

Table 3: Q-Former Subsystem Requirements and Verification

2.2.4 Large Language Model Subsystem

The Large Language Model (LLM) Subsystem is a sophisticated computational unit within our robotics architecture, integral for synthesizing both visual and textual data into actionable text outputs. It processes embeddings from the Image Encoder and Text Tokenizer Subsystems using an advanced **Llama** model, a choice inspired by the **BLIP-2** [1] architecture which ensures comprehensive and nuanced text generation. This subsystem's outputs are specifically formatted to instruct the Robot Operating System (ROS) for executing tasks or providing responses. The quantitative measure of this subsystem's performance is assessed by the quality and relevance of text outputs generated, as well as the speed and accuracy with which it processes input embeddings into these outputs. The interface with the Image Encoder and Text Tokenizer is marked by the standardized embedding vectors received, while its output interface with the ROS subsystem is quantified by the command strings dispatched for robotic control.

Requirements	Verification
1. Process visual and textual embeddings to generate accurate text outputs for robotic actions.	A. Test with predefined embeddings and compare outputs against expected action instructions. B. Evaluate output accuracy by conducting a series of blind tests with unknown embeddings.
2. Ensure compatibility with the BLIP-2 architecture to utilize the Llama model effectively.	A. Validate the subsystem's processing pipeline with the BLIP-2 specifications. B. Conduct integration tests to confirm seamless operation with BLIP-2 based systems.
3. Maintain a processing speed that allows real-time generation of responses and instructions.	A. Measure the time from embedding input to text output delivery. B. Benchmark the subsystem against real-time response requirements under varying load conditions.

Table 4: Large Language Model Subsystem Requirements and Verification

2.2.5 Text Tokenizer Subsystem

The Text Tokenizer Subsystem is a pivotal component tasked with converting raw textual inputs into structured embeddings. It serves as an intermediary, translating spoken language captured by the Speech-to-Text subsystem into a format amenable to computational analysis. Utilizing a **BERT** model tokenizer ensures compatibility with advanced Large Language Models like Llama, enabling robust text interpretation. This subsystem significantly contributes to the overall design by ensuring that linguistic information is accurately represented and processed, facilitating the system's ability to comprehend and act upon user commands. The interface with the Speech-to-Text subsystem is defined by the text input stream, while the output interface with the Large Language Model consists of tokenized text embeddings.

Requirements	Verification
1. Accurately tokenize text inputs using a BERT model tokenizer.	A. Validate tokenizer accuracy by comparing output against a known set of BERT model embeddings. B. Ensure tokenizer handles a variety of linguistic inputs without error.
2. Maintain processing latency within 50ms for converting text to embeddings.	A. Measure the time from text input reception to embedding output. B. Test across different text lengths to ensure consistent latency.
3. Interface seamlessly with the Speech-to-Text subsystem and the Large Language Model, maintaining data integrity.	A. Conduct end-to-end tests to verify data flow integrity from Speech-to-Text through to the Large Language Model. B. Check for any data loss or transformation issues during the interfacing.

Table 5: Text Tokenizer Subsystem Requirements and Verification

2.2.6 Speech-to-Text Subsystem

The Speech-to-Text Subsystem is a key interface that translates auditory information into a digital text format, bridging human interaction and machine processing. Using an advanced open-source model, this subsystem decodes spoken language with high accuracy and low latency, making it an essential component for real-time applications. It quantitatively contributes to the overall design by providing accurate text conversion, serving as the initial processing step for voice commands. The efficacy of this subsystem is measured by its transcription accuracy and speed, which directly impacts the performance of the downstream Text Tokenizer subsystem.

Requirements	Verification
1. Convert spoken language to text with a minimum accuracy of 95%.	A. Test the system with a predefined set of voice commands and compare the transcription to the expected text. B. Calculate the percentage of correctly transcribed words to assess accuracy.
2. Maintain transcription latency below 200ms to enable real-time processing.	A. Measure the time from speech input to text output using a stopwatch or relevant software tool. B. Perform repeated tests to ensure consistent latency below the 200ms threshold.
3. Ensure compatibility and seamless data transfer to the Text Tokenizer subsystem.	A. Verify the text output format is compatible with the Text Tokenizer's input requirements. B. Test the data transfer process between the subsystems for continuity and integrity of information.

Table 6: Speech-to-Text Subsystem Requirements and Verification

2.2.7 Microphone Subsystem

The Microphone Subsystem is an integral component of our robotic system, tasked with capturing audio input from users in a clear and reliable manner. Utilizing a Bluetooth microphone, this subsystem offers flexibility and enhances the robot's ability to interact with its environment by ensuring high-quality audio capture. This audio is then transmitted to the Speech-to-Text subsystem, where it is converted into textual data for further processing. The performance of this subsystem is quantitatively measured by its audio capture fidelity, noise reduction capability, and the latency in transmitting the captured audio to the Speech-to-Text subsystem. Its seamless integration and reliability are critical for the effective operation of the robot's interactive capabilities.

Requirements	Verification
1. Capture audio with high fidelity using a Bluetooth microphone.	A. Record various audio samples in different environmental conditions and analyze for clarity and fidelity. B. Conduct subjective listening tests to assess audio quality.
2. Ensure minimal transmission latency to the Speech-to-Text subsystem.	A. Measure the time from audio capture to receipt by the Speech-to-Text subsystem. B. Verify that this latency is within acceptable limits for real-time processing.
3. Maintain stable Bluetooth connectivity for audio capture.	A. Test the Bluetooth connection stability over various distances and with potential obstructions. B. Evaluate the system's performance in reconnecting automatically after any disconnections.

Table 7: Microphone Subsystem Requirements and Verification

2.2.8 ROS Subsystem

The ROS (Robot Operating System) Subsystem acts as the central control unit within our robotics framework, crucial for interfacing with both software components and hardware mechanisms. Leveraging MQTT, a lightweight messaging protocol, it facilitates real-time communication with the Raspberry Pi Auxiliary System, which manages the operational commands for the robot arm. This shift enhances the system's responsiveness and reliability, especially in low-bandwidth environments. The ROS Subsystem translates these commands into precise physical actions, coordinating closely with the Raspberry Pi to ensure seamless execution. Additionally, it continues to relay necessary responses back to the Text-to-Speech subsystem for user interaction. The system's performance is quantitatively assessed by its command execution latency, reliability in task execution, and the efficiency of inter-process communication.

Requirements	Verification
1. Execute robotic actions with minimal latency from the receipt of commands from the Raspberry Pi.	A. Time the interval from receiving commands via MQTT to the start of robot arm movement. B. Perform tests across various commands to ensure consistent low-latency response.
2. Ensure reliable inter-process communication using MQTT between the Raspberry Pi and the ROS system.	A. Test the stability and reliability of MQTT communication links under various network conditions. B. Conduct stress tests to evaluate the system's resilience and response accuracy using MQTT.
3. Maintain high reliability and efficiency in managing and executing tasks as directed by the Raspberry Pi.	A. Verify task execution fidelity and timing precision from multiple MQTT command sequences. B. Assess the overall system performance during extended operation periods to ensure sustained reliability.

Table 8: ROS Subsystem Requirements and Verification

2.2.9 Text-to-Speech Subsystem

The Text-to-Speech Subsystem is an essential communicative bridge in our robotics architecture, enabling the robot to convert textual responses into spoken words, thus facilitating a natural interaction with users. Utilizing the ***pyttsx3***[1], a versatile and open-source text-to-speech Python library, this subsystem translates textual data received from the ROS subsystem into audible speech, which is then relayed through the Voice Player subsystem for output. The choice of pyttsx3 not only supports a broad range of voices and languages but also ensures functionality without the need for internet connectivity. This subsystem's contribution to the overall design is quantitatively marked by its speech synthesis speed, clarity of the generated audio, and the seamless interface with the ROS and Voice Player subsystems, enabling the robot to provide timely and intelligible responses to user inquiries.

[1]https://pyttsx3.readthedocs.io/en/latest/

Requirements	Verification
1. Convert text outputs from the ROS subsystem into clear, audible speech.	A. Input known text strings to the subsystem and evaluate the clarity and audibility of the speech output. B. Test the subsystem with a variety of texts to ensure consistent performance across different speech patterns.
2. Maintain a conversion latency of less than 1 second for real-time communication.	A. Measure the time taken from text input to speech output and verify it falls within the 1-second threshold. B. Conduct latency tests under various system load conditions to assess performance stability.
3. Ensure compatibility with the Voice Player subsystem for effective audio playback.	A. Test the audio output format for compatibility with the Voice Player subsystem's playback requirements. B. Verify seamless integration through end-to-end testing from text input to speech playback.

Table 9: Text-to-Speech Subsystem Requirements and Verification

2.2.10 Voice Player Subsystem

The Voice Player Subsystem is a critical component for enabling the robot to audibly communicate with users, functioning as the final step in the interactive feedback loop. It takes the audio files generated by the Text-to-Speech Subsystem and plays them through a Bluetooth headset, ensuring clear and understandable speech output. This subsystem is essential for the robot's ability to provide audible responses to user queries or commands, enhancing the overall user experience. Its contribution to the design is quantified by its audio output clarity, playback latency, and compatibility with the Bluetooth headset, facilitating effective human-robot interaction.

Requirements	Verification
1. Play audio files generated by the Text-to-Speech subsystem with clear audio quality.	A. Test playback of various audio files to assess clarity and volume consistency. B. Conduct user feedback sessions to determine the intelligibility of the audio output.
2. Ensure minimal playback latency for real-time communication.	A. Measure the time delay from receiving an audio file to the start of playback. B. Verify that the latency does not exceed a predefined threshold, suitable for real-time interaction.
3. Maintain compatibility with Bluetooth headsets for audio playback.	A. Test connectivity and playback through various Bluetooth headsets to ensure universal compatibility. B. Evaluate the audio quality and connection stability across different headsets and distances.

Table 10: Voice Player Subsystem Requirements and Verification

2.2.11 Universal Robot UR3e Robot Arm Subsystem

The Universal Robot UR3e Robot Arm Subsystem, enhanced with a Makeblock Robot Gripper, is a critical component of our robotics architecture, providing high precision and flexibility for physical tasks. The gripper allows the arm to grasp and handle objects like water bottles more effectively. The gripper is controlled by a Raspberry Pi, which interfaces directly with the ROS subsystem to receive and execute detailed instructions. Equipped with six rotational joints, the subsystem executes movements with high precision, crucial for the accurate positioning and handling of objects within its operational environment. This subsystem's performance metrics include its reach, payload, repeatability, and the added functionality of the gripper's grasping capabilities.

Requirements	Verification
1. The UR3e, equipped with the Makeblock Gripper, must accurately follow Raspberry Pi and ROS instructions to grasp and move the water bottle.	A. Program the Raspberry Pi and ROS with a set of test instructions for grasping and moving, and observe if the robot arm with the gripper executes them correctly. B. Measure the repeatability of the arm's movements and the gripper's grasping actions between multiple test cycles.
2. The robot arm should have a reach and gripper capability sufficient to move between the designated area and the water dispenser.	A. Verify the arm's reach and gripper extension against the distance between the designated area and the water dispenser. B. Test the full range of motion of both the arm and the gripper to confirm no restrictions in the system's reach.
3. Ensure the UR3e operates within its standard operational speed during tasks, including the gripper actions.	A. Measure the time taken for the robot arm to complete a pick-and-place cycle including the time to grasp with the gripper. B. Compare the operational speed against the UR3e's and gripper's specified speed capabilities.

Table 11: UR3e Robot Arm Subsystem Requirements and Verification

2.2.12 Raspberry Pi Auxiliary Subsystem

The Raspberry Pi Auxiliary Subsystem serves as a sophisticated monitoring and interaction enhancer between the user and the robotic system, utilizing three cameras and a speaker for precise control and feedback. The first two cameras are dedicated to monitoring the water bottle on the desk, ensuring it is within the Robot Arm's reach and providing precise audio instructions for adjustments along the x-axis and y-axis, such as "Move your bottle right/left/forward/back a bit." The third camera checks the positioning of the bottle at the water dispenser to guarantee accurate filling. This subsystem leverages advanced object recognition technology to pinpoint the bottle's location and provide verbal guidance accordingly. Connected to the PCB Water Dispenser Subsystem, the Robot Gripper, and the ROS Subsystem, it orchestrates a seamless interaction flow and enhances operational efficiency. This integration is critical for the system's functionality, offering real-time and intuitive user guidance.

Requirements	Verification
1. Accurate detection of the water bottle's placement by both cameras.	A. Perform tests with the bottle placed in various positions to verify detection accuracy. B. Analyze the system's response time and accuracy in real-time conditions.
2. Clear and intelligible vocal instructions issued by the speaker.	A. Test the clarity and volume of vocal instructions in different environmental noise levels. B. Conduct user feedback sessions to assess the effectiveness of the instructions.
3. Integration with object recognition models to precisely identify the bottle's location.	A. Evaluate the object recognition accuracy through various placement tests. B. Verify the model's ability to adapt to different bottle shapes and colors.

Table 12: Raspberry Pi Auxiliary Subsystem Requirements and Verification

2.2.13 PCB Water Dispenser Subsystem

The PCB Water Dispenser Subsystem is integral to the functionality and user interface of our robotic system, acting as a visual communicator for the operational status of the water dispensing process. By employing a light control mechanism with programmable LEDs, this subsystem indicates when the water is being dispensed (green light) and when the process is complete (red light), based on the input from the Raspberry Pi Auxiliary System. This direct, visual feedback mechanism is crucial for coordinating the actions of the robot arm, especially in guiding it to retrieve and return the filled water bottle to the user. The subsystem's design is quantitatively defined by the accuracy of signal reception, the precision of the internal timer for light transitions, and the reliability of sending completion signals back to the Raspberry Pi Auxiliary System. This ensures seamless integration within the broader system, enhancing the robot's interactive capabilities.

Requirements	Verification
1. Accurate reception of bottle placement signals from the Raspberry Pi Auxiliary System.	A. Simulate signal transmissions from the Raspberry Pi Auxiliary System to verify accurate reception. B. Conduct repeated tests to ensure consistency in signal detection and processing.
2. Precise control of light indicators to reflect the dispensing status accurately.	A. Test the transition from green to red light after a predefined interval, verifying the internal timer's accuracy. B. Evaluate the visibility and clarity of light indications under various ambient lighting conditions.
3. Reliable communication of the completion signal back to the Raspberry Pi Auxiliary System.	A. Measure the reliability of sending completion signals back to the Raspberry Pi Auxiliary System after the red light is activated. B. Perform stress tests to assess the subsystem's performance in continuous operation.

Table 13: PCB Water Dispenser Subsystem Requirements and Verification

2.3 Tolerance Analysis

A key design consideration is the latency in data transfer, which is critical to real-time interaction and control. We meticulously assess the latency focusiing on two main channels: user to computer, and computer to server.

User to Computer Data Transfer Analysis: A pivotal design concern is the latency during Bluetooth transmission of captured images and audio from head-mounted cameras and headsets to the computer. Assuming an operational distance of approximately 10 meters, we utilize the following formula to estimate Bluetooth transmission latency:

$$\text{Latency} = \frac{\text{Data Size}}{\text{Transmission Speed}} + \text{Propagation Delay}$$

Data Size is the total size of the data to be transmitted, measured in bits. Transmission Speed is the rate at which data is transmitted, measured in bits per second (bps). Propagation Delay is the time it takes for the signal to travel from the source to the destination, which can be calculated as the distance divided by the speed of the signal. However, for Bluetooth and similar short-range technologies operating at the speed of light, this delay is negligible compared to other factors.

Given Bluetooth 4.0's capability of up to 25 Mbps in high-speed mode and considering an average data packet size (1MB for a captured image), we can estimate the latency:

$$\text{Latency} = \frac{1 \times 10^6 \times 8 \text{ bits}}{25 \times 10^6 \text{ bits/sec}} = 0.32 \text{ seconds}$$

Computer to Server Data Transfer Analysis: Through simulations, we have estimated that data transfer delays between the computer and server can be confined to approximately 3-4 seconds. This latency is primarily influenced by network speed, server processing capabilities, and the data's complexity. Incorporating Python libraries like flash-attention has been instrumental in augmenting our AI models' processing speeds. These libraries enable more efficient handling of computations necessary for real-time analysis and decision-making based on the data received from user devices.

Conclusion: Experimental outcomes demonstrate that, despite variations, the entire processing duration stays within a few seconds, contingent on the complexity of the input data. This duration falls within our acceptable limits for real-time operations, underscoring the system's viability for responsive and effective user assistance. This analysis confirms our commitment to optimizing system performance while maintaining the real-time interaction that is vital for the success of our project.

2.4 Cost and Schedule

2.4.1 Cost Analysis

Our fixed labor salary is estimated to be $10/hour, and 50 hours for each person. The total labor costs for all partners:

$$4 \cdot \$10/\text{hour} \cdot 2.5 \cdot 50 \text{ hours} = \$5000$$

The costs of all parts in our project are shown in Table 14.

Part	**Cost**
Personal Computer (Macbook)	$1200
Bluetooth Headset and iPhone Camera	$1000
Raspberry Pi System	$150
PCB Board (with Control Lights)	$20
Robot Arm and AI Server (Borrow from ZJUI)	$0

Table 14: Cost of Each Part

The grand total costs: $5000 + $1200 + $1000 + $150 + $20 = $7370.

2.4.2 Schedule

Already Done:

- Deployed Speech-to-Text system on computer.
- Deployed Text-to-Speech system on computer.
- Realized real-time file transmission between computer and server.
- Tested the Speech-to-Text and Text-to-Speech modules (translation was successful).
- Tested the server. Uploaded some videos to chat with the Large Vision Language model on the server.
- Tested the robot arm in ZJUI Lab E212. Set some basic joint coordinates to make it move. Made a simulation of moving the water bottle from one place to another.

Week 2024/3/25:

- **Haozhe Chi:** Try to reduce the delay in transmitting data between headset devices, personal computer, and the server.
- **Minghua Yang:** Buy Raspberry Pi and start to design the Raspberry Pi Auxiliary System.
- **Jiatong Li:** Test if the iPhone camera can transmit real-time high-quality images. This part is for the real-time visual inputs of the AI model.
- **Zonghai Jing:** Design and build circuits to indicate the working status of the water dispenser.

Week 2024/4/1:

- **Haozhe Chi:** Try to reduce the further delay in transmitting data between headset devices, personal computer, and the server.
- **Minghua Yang:** Design and build up the Raspberry Pi Auxiliary System.
- **Jiatong Li:** Test if both the iPhone camera and the Bluetooth headset can cooperate well and provide real-time guidance information.
- **Zonghai Jing & Jiatong Li:** Design the PCB board for controlling the water dispenser with the built circuits.

Week 2024/4/8:

- **Haozhe Chi:** Fix bugs in real-time questioning-answering tests and try to reduce further the delay in transmitting data between headset devices, personal computer, and the server.

- **Minghua Yang:** Develop the visual part of the Raspberry Pi system to detect the position of water bottle.
- **Jiatong Li:** Design and buy the PCB board for controlling the water dispenser.
- **Zonghai Jing:** Test if robot arm with more joints is available and help fix bugs in robot arm.

Week 2024/4/15:

- **Haozhe Chi:** Improve the speed of AI by adding accelerating components, such as flash-attention.
- **Minghua Yang:** Develop the audio part of the Raspberry Pi system to output voice instructions in order to help user place the water bottle.
- **Jiatong Li & Zonghai Jing:** Build up the PCB board (Soldering).
- **Zonghai Jing:** Help fix bugs in robot arm and make its movement smooth when carrying the water bottle.

Week 2024/4/22:

- **Haozhe Chi:** Improve the speed of Speech-to-Text module.
- **Minghua Yang:** Develop the control part of the Raspberry Pi system for communication with PCB board.
- **Jiatong Li:** Test the functionalities of PCB board.
- **Zonghai Jing:** Help fix bugs in robot arm and make its movement smooth when carrying the water bottles.

Week 2024/4/29:

- **Haozhe Chi:** Improve the speed of Text-to-Speech module.
- **Minghua Yang:** Combine the visual and audio part of Raspberry Pi system together, perform testing.
- **Jiatong Li & Minghua Yang:** Connect the Raspberry Pi system and the PCB board together and set up communication. Raspberry Pi system should control the PCB board.
- **Zonghai Jing:** Help fix bugs in robot arm and make its movement smooth when carrying the water bottles.

Week 2024/5/6:

- **Haozhe Chi & Jiatong Li:** Test if all the components for visual language navigation work well, including the headset devices, personal computer, and the AI server. Fix bugs if needed.
- **Minghua Yang & Jiatong Li:** Test if the Raspberry Pi system can control the PCB to indicate the working status of the water dispenser.
- **Zonghai Jing & Minghua Yang:** Test if the Raspberry Pi system can provide voice instructions to help user place the water bottle, if the robot arm can move the bottle smoothly and work as expected.

Week 2024/5/13: Integrate each part together and prepare for the final presentation.

3 Ethics and Safety

In the development of our robotics project, we rigorously adhere to the IEEE Code of Ethics [2] to uphold the highest standards of ethical practice and safety.

3.1 Ethics

Privacy (ACM 1.7: Respect the Privacy of Others) To protect privacy, we take strict measures to ensure the confidentiality and security of any personal information collected by the robot. We establish robust protocols based on industry standards to limit access to this sensitive data to authorized individuals with a legitimate need to access it. In addition, we carefully design and implement secure storage and processing procedures to reduce the risk of unauthorized disclosure or misuse. By prioritizing the protection of personal information, we demonstrate our unwavering commitment to maintaining the privacy and trust of individuals who interact with our robots.

Fairness (IEEE - Avoiding Real or Perceived Conflicts of Interest) The possibility of bias in the decision-making process of artificial intelligence is a major ethical issue. Recognizing this, we will strive to provide robots with a comprehensive understanding of human diversity and societal nuances through rigorous training and careful refinement. By exposing robots to a variety of data, including different demographics, cultural backgrounds, and environmental scenarios, we aim to equip robots with the ability to impartially discern and understand complex social dynamics.

Being Open (ACM 1.2: Avoid Harm) We are committed to ensuring full transparency in the robotics decision-making process. Our goal is to provide clear and understandable information to all stakeholders so that they can fully understand how the robot operates and the factors that influence its decisions. To achieve this, we keep detailed records of the algorithms, data inputs and learning methods used by the robot. Additionally, we are committed to an open approach to making information about the robot's functioning,

including its training data, learning outcomes, and decision logic, readily available. By increasing transparency, we aim to build trust and confidence among users, stakeholders, and the broader community, thereby promoting ethical behavior by individuals or organizations when using our robotics.

Professional Development (ACM 2.6) Adhering to ACM's principles, our team dedicates itself to the continual enhancement of our knowledge and understanding of the societal ramifications of robotics. We recognize the dynamic nature of ethical standards and proactively refine our systems to stay abreast of new developments, ensuring that our robots serve as a benchmark for responsible AI and robotics practice.

3.2 Safety

Avoiding Accidents (IEEE - Priority to Public Welfare) Our robots are carefully designed with safety as a top priority to ensure that they do not jeopardize the personal safety of others or the safety of property. Equipped with advanced emergency stops and a range of sophisticated sensors, the robots are able to operate with increased vigilance, effectively preventing collisions with people and objects. These safety features are carefully designed to prevent accidental collisions and provide peace of mind in dynamic environments where human-robot interactions are frequent.

Staying Secure (ACM 3.7: Recognize the Need to Protect Personal Data) Given the advanced functionality and interconnectedness of our robots, it is critical to protect their integrity and guard against potential cyber threats. We are therefore building relevant security measures to strengthen its defenses and reduce the risks posed by malicious actors and cyberattacks. This requires the implementation of advanced encryption protocols, strict access controls and continuous monitoring mechanisms to detect and respond to any unauthorized attempts to compromise robotic systems or data. In addition, we prioritize regular security assessments and audits to identify vulnerabilities and weaknesses in our security infrastructure, enabling us to proactively address potential threats and ensure that our robots are resilient to evolving cyber threats.

Dealing with Mistakes (ACM 2.5 & IEEE - Acknowledge and Correct Mistakes) In the event of an unforeseen situation or error, the robot responds in a manner that prioritizes safety and reliability. The robot's operational framework incorporates fail-safe mechanisms and real-time monitoring capabilities to promptly identify and address any anomalies or deviations from expected behavior. By promptly notifying designated personnel or stakeholders of such occurrences, the robot facilitates rapid intervention to minimize potential risks and ensure continuity of safe and effective operations.

Responsibility (IEEE) In line with IEEE guidelines, our project is committed to the responsible deployment of robotics, ensuring they fulfill their intended roles effectively

while safeguarding societal and environmental well-being. Our team maintains a vigilant approach to technology stewardship, regularly assessing and mitigating any negative impacts our robots may have, thereby ensuring our innovations contribute positively to society and operate sustainably within the environment.

Whistleblowing (ACM 1.4) Upholding ACM's ethical code, we foster an environment where whistleblowing is not just protected but encouraged, as it is crucial for maintaining the highest ethical standards. By promoting transparency and inviting scrutiny, we ensure any instance of misuse or ethical misconduct involving our robots is promptly addressed, reinforcing our commitment to integrity and the responsible use of technology.

References

[1] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," *arXiv preprint arXiv:2301.12597*, 2023.

[2] Institute of Electrical and Electronics Engineers. (2016) IEEE Code of Ethics. Accessed on: 2024-03-07. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html