Proceedings of the 3rd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/138/2025.21317

Home Assist Robots via Vision and Oral Guidance

Cindy Xiao[1,a,*]

[1]*Monta Vista High School, 21840 McClellan Rd, Cupertino, CA 95014, US*
a. cindy.xiao.ca@gmail.com
**corresponding author*

Abstract: With the acceleration of population aging, the demand for elderly care services is growing rapidly, and the shortage of caregivers has become a global social issue. To address this challenge, the application of artificial intelligence technology in the field of elderly care, and the development of assistive robots capable of providing personalized and efficient care services, has become a focus of research. This study aims to develop an intelligent robot that can understand elderly people's instructions, perceive the surrounding environment, and provide personalized assistance services by integrating advanced artificial intelligence technologies such as image recognition and natural language processing into the robotic system. Specifically, the robot will have the following functions: First, the robot can conduct natural language interaction. Through speech recognition and semantic understanding technology, it can accurately recognize the elderly person's voice commands and conduct natural and fluent conversations. Second, the robot can realize visual perception. By utilizing image recognition technology, it can perceive the surrounding environment in real-time, recognize objects, faces, etc., and respond accordingly based on the scene. Finally, based on the understanding of instructions and the perception of the environment, the robot can complete a series of daily tasks such as fetching objects, companionship, and reminders. By developing and evaluating such intelligent robots, we hope to improve the quality of life for the elderly, reduce the workload of caregivers, promote innovation in elderly care service models, and provide technical support and theoretical foundation for future intelligent elderly care.

Keywords: Ros, Robot, Vision, Home Assist.

1. Introduction

According to the California Health Care Foundation, the aging population in many states such as California will require an increase in support from direct care workers to care for their essential needs,yet experts predict a shortage of between 600,000 and 3.2 million direct care workers by 2030. They need special care but cannot afford human labor. Because of this severe labor shortage problem,introducing home service robots is a more cost-efficient way to solve this problem.

In a publication titled ”Service Robots in the Healthcare Sector,” Jane Holland and others have researched and developed service robots in the healthcare sector, and found different uses for robots and their designs and flow for implementation. Furthermore, in the article ”Robotics in Healthcare: The Future of Robots in Medicine” by Intel talks about reducing the workload on health care providers by using a systematic and technical routine through programming. It can also help engage with and

DOI: 10.54254/2755-2721/138/2025.21317

care for patients. However, these robots are mainly for surgical-assistance. Moreover, the Association for Advancing Automation Robotics strive to improve the quality of healthcare by creating diverse medical robots. Their robots include the telepresence robot for remote care-giving, disinfectant robots to reduce hospital-acquired infections, robots that can more accurately and efficiently draw blood,and robotic exoskeletons that provide external support and muscle training for rehabilitation. Mobile medical robots are also being used for delivery of medication and other sensitive materials in a hospital setting.

Future areas of research need to be discussed, specifically focusing on short, medium and long-term technology directions in an effort to emphasize the need to improve current service robotics applications.

I have created a robot that assists the elderly or those who are disabled with a home robot that can aid the user through its vision and oral guidance.

2. The System Overview

The following diagram illustrates a network configuration for controlling a robotic arm through visual and voice interfaces. A detailed description of each component and its connections is provided below:

Internet Router: The system begins with an internet router providing Wi-Fi connectivity.

Wi-Fi Extender: This device extends the Wi-Fi signal, connecting to the main internet router via Wi-Fi and to the computer wirelessly.

Computer: Running Ubuntu Linux, the computer connects to the Wi-Fi extender wirelessly. Additionally, a microphone is connected to the computer, indicating a voice interface for voice commands.

Robotic Arm: The robot utilizes a Raspberry Pi as its control device, running Ubuntu 18.04 and ROS. The robotic arm and the computer communicate via ROS in a master-slave relationship. The Raspberry Pi receives instructions from the computer and controls the actual movement of the robotic arm. Furthermore, the robotic arm is equipped with a vision system (such as a camera) for object detection and navigation.

In this project, colors are used to simulate tasks that elderly people need a robot to complete in daily life. The robot's line-following movement simulates the process of the robot accomplishing the received instructions. In the actual implementation, the tester issues voice commands such as "red," "yellow," "blue," or "green" on the host computer. The host computer converts the voice into text information using ros_speech_recognition and publishes it to the Speech-to-Text topic. A ROS node subscribes to the topic message, transforms it into a message type that the lower-level system can recognize, and sends it to a new topic. The robot and host computer communicate using ROS in a master-slave relationship, and the robot subscribes to the command topic. Then, based on the color command in the topic, the robot performs line-following movement on the corresponding color line.

Overall, Linux enables human-computer interaction through voice commands. Linux and Raspberry Pi communicate via ROS topics. The Raspberry Pi sends commands to control the movement of the robotic arm, and the vision module enables interaction between the robotic arm and the environment.

DOI: 10.54254/2755-2721/138/2025.21317

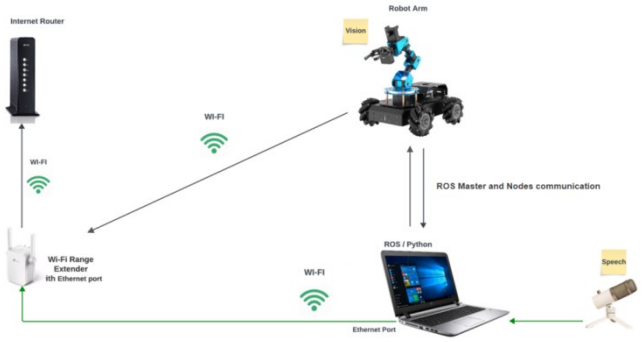

Figure 1: The System Overview

3. Speech-to-text subsystem

The goal of speech recognition technology is to convert human speech signals into text or commands that can be understood and processed by computers. This process involves several key steps: speech acquisition, signal processing, feature extraction, acoustic modeling, language modeling, and decoding.

First, we capture speech signals through a microphone or other devices and convert them into digital format. Next, these digital signals undergo pre-processing to reduce noise and improve signal quality. Then, useful features are extracted from the processed signals, which effectively represent the acoustic characteristics of speech.

Subsequently, machine learning algorithms are used to train acoustic models that match the extracted features with the sound units (such as phonemes) in a language. To improve speech recognition accuracy, we also use language models to predict and identify word sequences. Language models can be based on statistical probabilities or deep learning methods.

Finally, we combine the outputs of the acoustic and language models to perform decoding and generate the final text or commands. Modern speech recognition technology commonly employs deep learning algorithms, such as neural networks, to enhance recognition accuracy and processing capabilities. Applications are widespread, including virtual assistants (like Siri, Alexa), automatic subtitle generation, and voice-controlled devices.

In this project, we utilize the ROS package ros_speech_recognition [1], which employs the Python package SpeechRecognition [2] as its backend. For the specific implementation, we use the Google Cloud Speech Library as the speech recognition engine.

3.1. Ros_speech_recognition

Ros_speech_recognition is a ROS [3] package specifically designed to convert speech signals into textual information that can be understood by the ROS system. This package communicates with other components within the ROS system through ROS nodes and topics. It supports multiple speech recognition engines, allowing developers to choose the most suitable engine for their project. ros_speech_recognition encapsulates the recognized speech text into ROS messages and publishes them. Other ROS nodes can subscribe to these messages and perform corresponding actions based on the message content. This design makes ros_speech_recognition an ideal choice for building voice-controlled robots.

DOI: 10.54254/2755-2721/138/2025.21317

3.2. Speech Recognition

Speech Recognition is a robust Python library specifically designed to transcribe audio into text. It supports a variety of popular speech recognition engines and APIs, offering developers flexibility in their choice. It can be used to build voice assistants, transcribe audio files, or perform speech analysis.

Speech Recognition supports several well-known speech recognition engines such as Google, Microsoft Azure, IBM Watson, and Sphinx. It can run on multiple operating systems including Windows, macOS, and Linux, and supports various audio formats like WAV, FLAC, and MP3. The library provides an intuitive API interface and allows users to set various parameters such as sample rate and number of channels. Through the Sphinx engine, it enables offline speech recognition without an internet connection.

3.3. Google Cloud Speech Library [4]

Google Cloud Speech-to-Text is a cloud-based service provided by Google Cloud that accurately transcribes audio into text. Whether it's a meeting recording, a voicemail, or the audio from a video, Speech-to-Text can effortlessly convert these audio files into searchable and editable text.

Google Cloud Speech-to-Text is a powerful speech recognition tool that enables real-time transcription and batch audio processing. Whether it's a real-time audio stream or an audio file stored in the cloud, Speech-to-Text can efficiently convert it into text. Additionally, it supports multiple languages and dialects and has robust noise suppression and speaker diarization capabilities. With its voice activity detection feature, Speech-to-Text can accurately identify speech segments, providing users with more precise transcription results.

Google Cloud Speech-to-Text employs an innovative speech recognition approach centered around Chirp, a universal speech model trained on massive, multilingual self-supervised data. Unlike traditional speech recognition systems that heavily rely on large amounts of language-specific supervised data, the Chirp model achieves a more powerful language modeling capability through self-supervised learning on millions of hours of audio data and billions of text sentences. This approach enables Speech-to-Text to more accurately recognize various languages and dialects and exhibit greater robustness to accents and noise.

To further improve recognition accuracy and user experience, Speech-to-Text introduces model adaptation techniques. This technology allows users to customize the model for specific application needs, such as adding domain-specific vocabulary or adjusting the model's sensitivity to certain phonemes. This flexible customization capability enables Speech-to-Text to better adapt to the specific needs of different industries.

3.4. Implementation

1. Install ROS and Speech Recognition Library: Install the corresponding ROS package based on the chosen speech recognition engine, Google Speech-to-Text.

2. Create a ROS Node: Create a ROS node that subscribes to an audio data topic. Within the node, use the selected speech recognition engine to process the received audio and publish the recognition results to a new topic.

3. Configure Parameters: Configure parameters according to the Google Speech-to-Text engine, such as language model, acoustic model, and sampling rate.

4. Test and Debug: Visualize and debug the node's operation using ROS tools. Record an audio clip as input to test the accuracy of the speech recognition.

DOI: 10.54254/2755-2721/138/2025.21317

4. PC-to-ROBOT communication

ROS employs a distributed communication model that enables different functional modules, or nodes, to collaborate. This flexible architecture allows for easy expansion and customization of robotic systems. The core of ROS communication is message passing between nodes. Each node operates as an independent process, responsible for a specific task. They discover each other and establish communication connections through a central coordinator, the ROS Master. The ROS Master acts like a phonebook, recording the contact information of all nodes in the system, enabling nodes to easily locate and communicate with each other.

ROS offers a variety of communication mechanisms to meet different needs.

Topics: This is a publish/subscribe pattern, similar to broadcasting. Nodes can publish data to a topic, and other interested nodes can subscribe to that topic to receive data in real-time. For example, a node can publish sensor data to a topic named "/sensor_data," while another node subscribes to this topic to obtain the sensor data for processing.

Services: This is a request/response pattern, similar to function calls. A node can send a service request to another node, and the requested node will process the request and return a result. For example, a node can send a "navigate to goal" request to a navigation node, and the navigation node will calculate the path and return the result.

Actions: This is an extension of the service pattern, suitable for long-running tasks that require feedback. For example, a robot arm control node can execute a "grasp object" action, providing continuous feedback on the grasping progress during execution.

The communication process between nodes is generally as follows: a node first registers itself with the ROS Master and declares the services it provides or the topics it subscribes to. Other nodes can query the ROS Master to find the required nodes and establish direct communication connections. Data is transmitted between nodes using the TCP/IP protocol.

The advantages of ROS communication lie in its flexibility and scalability. Each node can be developed and tested independently, and the system can add or remove nodes as needed. Additionally, ROS promotes loose coupling: nodes communicate through topics, reducing coupling between nodes and improving system maintainability. Furthermore, ROS offers real-time capabilities, providing various tools for real-time data transmission and synchronization.

In summary, the master-slave communication mechanism of ROS provides a robust communication foundation for robotic systems, enabling different functional modules to collaborate and achieve complex robotic behaviors. Through flexible communication methods and a distributed architecture, ROS has become the preferred platform for robotics research and development.

4.1. Implementation

Adding IP Addresses and Setting up ROS Master

I. Adding IP Addresses

Ensure both the host and slave machines are on the same local network. Use the ifconfig command to check the IP addresses of the host and slave, and use the hostname command to check their hostnames.

In the /etc/hosts file, add the slave's IP address and hostname to the host, and vice versa. Once this is done, the host and slave should be able to ping each other.

II. Setting up ROS Master

Since only one machine can serve as the ROS Master in ROS1, we need to set the ROS_MASTER_URI for all slave machines in their .bashrc files.

With these steps completed, the communication between the host and slave machines within the local network is configured.

DOI: 10.54254/2755-2721/138/2025.21317

5. Robot line following vision system

5.1. Mechanism

Initially, color recognition is essential. We employ the Lab color space for this purpose. By first converting the RGB color space to Lab and subsequently applying binarization, dilation, and erosion operations, we can isolate contours of the target color. The contours of these lines are then enclosed within bounding boxes, and the centroids of these lines are marked within these boxes.

Upon successful identification of the lines corresponding to the specified color, the ArmPi Pro robot proceeds to follow the lines, thus achieving intelligent line-following functionality.

5.2. Image processing[5]

1. Image binarization

Employ the inRange() function provided by the OpenCV library (cv2) to perform binary thresholding on the image.

2. Erosion and dilation

Morphological operations, including erosion and dilation, are employed to denoise and smoothen the image.

3. Find the contour with the largest area.

After completing the image processing, we need to obtain the contour of the recognized target. The findContours() function in the OpenCV library (cv2) is used for this purpose.

4. Obtain location information.

We utilize the minAreaRect() function from the OpenCV library (cv2) to obtain the minimum area bounding rectangle of the target contour. The boxPoints() function is then employed to acquire the coordinates of the four vertices of this rectangle. Subsequently, the center coordinates of the rectangle can be calculated based on these vertex coordinates.

5. Line tracking control[6]

Following image processing, the velocity of the ArmPi Pro robot's motors is set through the invocation of the set_velocity.publis() function, thereby controlling its movement.

6. Optimization scheme

In reality, the needs of the elderly for robots are diverse. Simple color recognition is insufficient to address more complex real-world scenarios. For instance, the elderly may require robots to perform tasks such as retrieving specific objects or recognizing routes. Considering these factors, we aim to improve the existing system by adding the robot's ability to recognize real-world objects.

6.1. Datasets

I used the data set from https://universe.roboflow.com/roboflow-100/furniture-ngpea, which had different categories of furniture including sofas, tables, and chairs. The different categories of furniture are used to train an image recognition model that helps the robot detect what type of furniture is ahead of it.

6.2. The finetuning network from ResNet

Fine-tuning a network like ResNet (Residual Network) involves adapting a pre-trained model to a specific task, typically by adjusting its weights and potentially modifying its architecture. ResNet is a deep convolutional neural network (CNN[7]) known for its ability to train very deep networks by introducing residual connections, which help mitigate the vanishing gradient problem.

DOI: 10.54254/2755-2721/138/2025.21317

There are two major approaches for finetuning. The first approach is using the model as it is and then just finetuning it on a different dataset. The second approach is using the pre-trained network as a feature extractor, and then changing the later layers of the pretrained model, and finetuning it. My work belongs to the second category.

The modification I have done in finetuning is making it be labeled to do a specific task.

6.3. The finetuning performance

The original accuracy of resnet in our data before finetuning is 0.7815, for epoch 0.

'train': 453, 'val': 160

Epoch 0/24 ——- train Loss: 0.6026 Acc: 0.7815 val Loss: 0.1710 Acc: 0.9625
Epoch 1/24 ——- train Loss: 0.4979 Acc: 0.8499 val Loss: 0.0625 Acc: 0.9750
Epoch 2/24 ——- train Loss: 0.3587 Acc: 0.8698 val Loss: 0.1481 Acc: 0.9500
Epoch 3/24 ——- train Loss: 0.3606 Acc: 0.8720 val Loss: 0.0735 Acc: 0.9812
Epoch 4/24 ——- train Loss: 0.2989 Acc: 0.9051 val Loss: 0.1562 Acc: 0.9375
Epoch 5/24 ——- train Loss: 0.4158 Acc: 0.8521 val Loss: 0.2573 Acc: 0.9250
Epoch 6/24 ——- train Loss: 0.3851 Acc: 0.8786 val Loss: 0.0503 Acc: 0.9938
Epoch 7/24 ——- train Loss: 0.3153 Acc: 0.8962 val Loss: 0.0437 Acc: 0.9875
Epoch 8/24 ——- train Loss: 0.1885 Acc: 0.9360 val Loss: 0.0446 Acc: 0.9938
Epoch 9/24 ——- train Loss: 0.2318 Acc: 0.9095 val Loss: 0.0488 Acc: 0.9938
Epoch 10/24 ——- train Loss: 0.2090 Acc: 0.9095 val Loss: 0.0389 Acc: 0.9875
Epoch 11/24 ——- train Loss: 0.1986 Acc: 0.9294 val Loss: 0.0600 Acc: 0.9875
Epoch 12/24 ——- train Loss: 0.2366 Acc: 0.9095 val Loss: 0.0595 Acc: 0.9812
Epoch 13/24 ——- train Loss: 0.2101 Acc: 0.9227 val Loss: 0.0535 Acc: 0.9938
Epoch 14/24 ——- train Loss: 0.1277 Acc: 0.9603 val Loss: 0.0446 Acc: 0.9938
Epoch 15/24 ——- train Loss: 0.1337 Acc: 0.9448 val Loss: 0.0445 Acc: 0.9875
Epoch 16/24 ——- train Loss: 0.1478 Acc: 0.9448 val Loss: 0.0573 Acc: 0.9938
Epoch 17/24 ——- train Loss: 0.2183 Acc: 0.9272 val Loss: 0.1079 Acc: 0.9750
Epoch 18/24 ——- train Loss: 0.1933 Acc: 0.9338 val Loss: 0.0513 Acc: 0.9812
Epoch 19/24 ——- train Loss: 0.1590 Acc: 0.9492 val Loss: 0.0376 Acc: 0.9938
Epoch 20/24 ——- train Loss: 0.1775 Acc: 0.9426 val Loss: 0.0438 Acc: 0.9938
Epoch 21/24 ——- train Loss: 0.1758 Acc: 0.9360 val Loss: 0.0439 Acc: 0.9938
Epoch 22/24 ——- train Loss: 0.1447 Acc: 0.9625 val Loss: 0.0531 Acc: 0.9812
Epoch 23/24 ——- train Loss: 0.1654 Acc: 0.9448 val Loss: 0.0876 Acc: 0.9750
Epoch 24/24 ——- train Loss: 0.1717 Acc: 0.9382 val Loss: 0.0533 Acc: 0.9812
Training complete in 12m 28s Best val Acc: 0.993750

The accuracy of the finetuned resnet on our datasets is 0.993750. It is more accurate than the original one. 1) The performance on dataset I; The best accuracy number is 0.993750.

7. Project performance showcase

7.1. Audio input from microphone

The microphone is deemed operational if it exhibits real-time volume fluctuations in response to audio input.

DOI: 10.54254/2755-2721/138/2025.21317

Figure 2: Microphone input quality

7.2. Speech Recognition

Steps to Initiate Speech Recognition Node and Test Functionality

1. Verify Capture Node Activation:

2. Launch Speech Recognition Node:

Activate the speech recognition node within your workspace's ros_speech_recognition package. Note: Prior to execution, verify that your workspace is correctly sourced into your ROS environment.

3. Confirm Speech Engine:

Double-check that the specified speech engine aligns with your chosen engine.

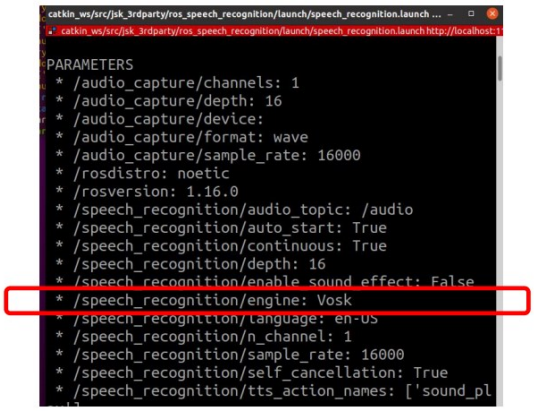

Figure 3: parameters

Google is the recommended platform for this project.

4. Speech Recognition Verification

To validate the speech recognition functionality, please articulate directly into the microphone. The application should accurately recognize and display the spoken words.

5. Inspecting Published Topic Data

This will enable you to scrutinize the specific topic data being disseminated by the speech recognition node, providing granular insights into the recognized speech content.

7.3. PC-to-ROBOT communication

Upon successful configuration of the master-slave communication:

If the listed topics include those initiated by the ArmPi FPV robotic arm, it signifies that the network configuration has been successfully established.

DOI: 10.54254/2755-2721/138/2025.21317

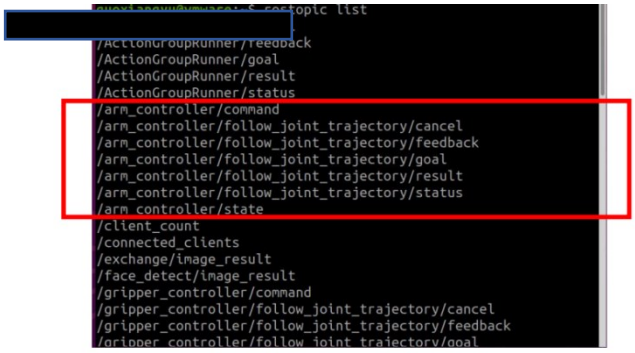

Figure 4: ropic list

7.4. To verify the robotic arm's ability to subscribe to topics disseminated by the host computer.

1. Initiating Speech Recognition Program on Virtual Machine

2. Verifying Topic Subscription on Raspberry Pi

The successful appearance of the topic /speech_to_text within the enumerated topic list signifies that the robotic arm has successfully subscribed to the speech recognition topic disseminated by the host computer.

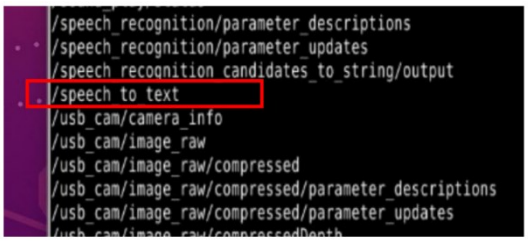

Figure 5: ropic list

7.5. Node input process

1. Executing the Startup Script for Robotic Arm To initiate the robotic arm, navigate to the specified directory and execute the startup script.

2. Voice Command for Color-Based Line Following Subsequently, issue voice commands such as "red," "blue," or "green" into the microphone.

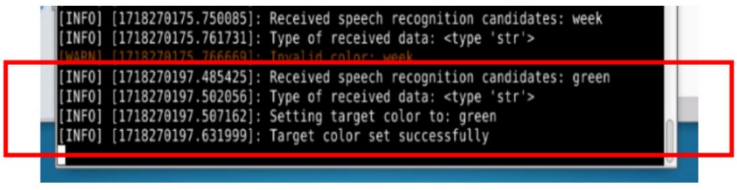

Figure 6: Input parameters and output results

Successful recognition of these commands will result in the robotic arm being configured to follow a line of the corresponding color.

DOI: 10.54254/2755-2721/138/2025.21317

7.6. Visual display of line-following motion[8]

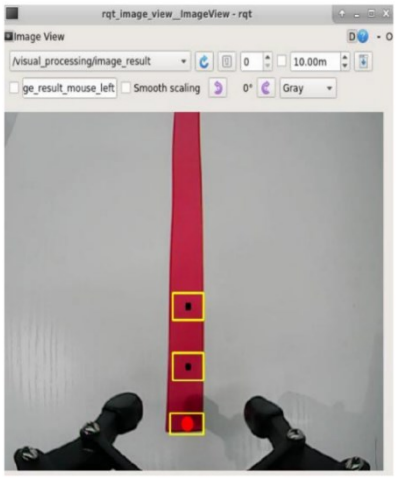

Figure 7: Visual result of line-following

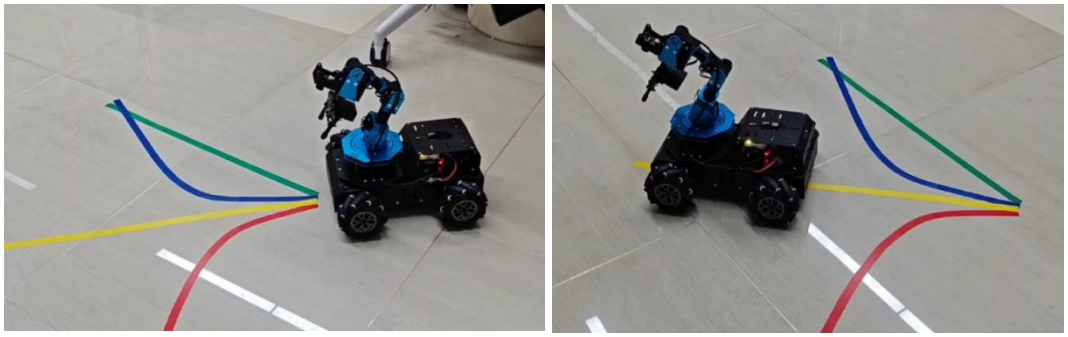

Figure 8: When the voice command is "yellow", the robot performs line-following based on the command.

8. Conclusion

Through our meticulous design, the robotic arm system has successfully achieved natural interaction with humans. Specifically, users can input voice commands to inform the robotic arm of the target color. The system receives these voice commands through a host computer, which then utilizes advanced speech recognition technology to accurately interpret the input. The parsed color command is subsequently transmitted to the robotic arm's control system, which precisely recognizes the instruction and moves along the path corresponding to the specified color. Experimental results demonstrate that the device performs exceptionally well, responding smoothly to and executing the user's voice commands, thereby realizing the expected functionality of controlling the robotic arm to perform tasks based on voice instructions.

This innovative design not only validates the practical application potential of speech recognition technology in the field of intelligent robotics, but also provides significant technical support for the development of elderly companion robot systems. By enabling intelligent interaction between the robotic arm and the user, our work opens new avenues for the advancement of smart elderly care services, particularly in enhancing the quality of life for the elderly and alleviating caregiver burden, thus demonstrating its vast application prospects and practical significance.

DOI: 10.54254/2755-2721/138/2025.21317

References

[1] JSK Robotics. (n.d.). ROS Speech Recognition: jsk_3rdparty. ROS Index. Retrieved November 10, 2024, from https://index.ros.org/p/ros_speech_recognition/github-jsk-ros-pkg-jsk_3rdparty/#noetic

[2] Phantom, S. (n.d.). SpeechRecognition. PyPI. Retrieved December 15, 2024, from https://pypi.org/project/SpeechRecognition/

[3] ROS.org. (n.d.). ROS Index. Retrieved January 15, 2025, from https://index.ros.org/

[4] Google Cloud. (n.d.). Speech-to-Text. Google Cloud. Retrieved January 10, 2025, from https://cloud.google.com/speech-to-text?hl=zh-cn

[5] Kovásznay L S G, Joseph H M. Image processing[J]. Proceedings of the IRE, 1955, 43(5): 560-570.

[6] Balaji V, Balaji M, Chandrasekaran M, et al. Optimization of PID control for high speed line tracking robots[J]. Procedia Computer Science, 2015, 76: 147-154.

[7] Alzubaidi L, Zhang J, Humaidi A J, et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions[J]. Journal of big Data, 2021, 8: 1-74.

[8] JSK Robotics. (n.d.). opt_camera. ROS Index. Retrieved January 17, 2025, from https://index.ros.org/p/opt_camera/github-jsk-ros-pkg-jsk_3rdparty/