2024 International Conference “ROBOTICS & MEHATRONICS”
29 – 30 October, 2024, Sofia, Bulgaria

Voice Controlled Interface for ROS Service Robot in Healthcare

Georgi Angelov
Institute of Robotics
Bulgarian Academy of Sciences
Sofia, Bulgaria
george@robotics.bg

Yasen Paunski
Institute of Robotics
Bulgarian Academy of Sciences
Sofia, Bulgaria
y.paunsky@ir.bas.bg

Abstract—**This paper presents the development of a voice-controlled interface for ROS-based service robots designed to support healthcare environments. By enabling hands-free, intuitive interaction through speech, this interface allows healthcare professionals to command robots without physical contact, a critical advantage in contaminated or infectious settings where minimizing touchpoints can significantly reduce cross-contamination risks. The system leverages advanced speech-to-text models, such as Whisper, integrated with ROS middleware, to interpret verbal commands accurately, even in noisy hospital environments. As voice recognition and real-time processing technologies evolve, such systems are expected to play an essential role in advancing operational effectiveness and patient care quality in medical settings.**

Keywords—voice user interface, ROS, service robots, VUI, UX, voice control

I. Introduction

Voice-controlled interfaces are an intuitive and hands-free method for operating service robots, making them particularly valuable in healthcare where staff frequently work in demanding, hands-on environments. By allowing healthcare workers to interact with robots using natural language, these interfaces simplify control, minimize physical effort, and help improve efficiency in a wide range of hospital tasks, from delivering medications to assisting with room sanitization. As the demand for automation in healthcare grows, voice-controlled robots provides a practical solution to streamline operations, reduce human workload, and support high-quality patient care.

The integration of a voice-controlled interface with ROS—the widely adopted middleware for robot programming and control—enables robust communication and coordination between different robotic components in a healthcare environment.

ROS facilitates modularity, allowing developers to build systems that can interpret and execute voice commands through multiple nodes for tasks such as navigation, object recognition, and manipulation. Using ROS, a voice-controlled service robot can interpret verbal commands to move between locations, fetch supplies, or adjust its actions based on real-time feedback, adapting to the dynamic nature of healthcare facilities.

II. Interfaces for Controlling ROS based Robots

A. Robot Operating System

The Robot Operating System (ROS) is a flexible framework for writing robot software, widely used for developing and controlling robots across various applications, from research and education to industrial automation and service robotics. As robots become more sophisticated, there is an increasing need for user-friendly web interfaces that allow operators to control and monitor robots running on ROS remotely. This chapter explores how web interfaces can be integrated into a ROS environment, the technologies commonly used for this purpose, and the advantages and challenges of this approach.

ROS provides a modular architecture where different software components, called nodes, communicate with each other through a messaging system. This enables developers to create complex robot behaviors by integrating various sensors, actuators, and processing algorithms. Web interfaces can be designed to interact with these ROS nodes, providing a graphical interface for users to issue commands, monitor real-time data, and manage robot configurations from a web browser.

The integration of web interfaces with ROS can be achieved using tools like rosbridge and ROS Web Tools, which provide a bridge between the ROS environment and web-based applications. This allows for seamless communication, where web clients can send commands to ROS nodes or receive data streams for visualization.

B. Key Components of Web Interfaces in ROS

- **rosbridge Suite**

The rosbridge suite is a set of packages that allows developers to create web interfaces for ROS. It provides a WebSocket-based protocol, enabling real-time communication between web clients (browsers) and ROS nodes. This means that users can interact with ROS topics, services, and parameters through web browsers without the need for additional client software.

- **roslibjs and roslibpy**

roslibjs is a JavaScript library that works with rosbridge to allow web applications to interact with ROS nodes. Developers can use roslibjs to send and receive messages,

ACKNOWLEDGE THE FINANCIAL SUPPORT OF THE PROJECT WITH ADMINISTRATIVE CONTRACT № KP-06-H57/8 FROM 16.11.2021. - FUNDED BY THE "COMPETITION FOR FUNDING BASIC RESEARCH - 2021." FROM THE RESEARCH SCIENCES FUND, BULGARIA.

call services, and manage parameters directly from a web page.

roslibpy is a similar library written in Python, enabling Python-based web servers to communicate with ROS systems. It can be useful for integrating ROS control with backend web frameworks like Flask or Django.

- **ROS Web Tools (e.g., RvizWeb, Web Video Server)**

RvizWeb is a web-based version of Rviz, a popular ROS tool for visualizing robot models, sensor data, and environments. It allows users to see real-time 3D visualizations directly in a web browser, which is useful for remote robot monitoring and diagnostics.

Web Video Server provides streaming of video feeds from robots' cameras to a web interface. This can be essential for teleoperation, where users need visual feedback to control the robot accurately.

All these tools enhance the functionality of web interfaces by providing critical visual data, making it easier to monitor and control robots remotely. They also enable the development of more sophisticated interfaces that offer real-time visual feedback.

C. Development Technologies for Web Interfaces in ROS

1. JavaScript and Web Frameworks (React, Vue.js, Angular)

JavaScript, along with modern web frameworks like React, Vue.js, and Angular, is commonly used to develop the front-end of web interfaces for ROS. These frameworks provide tools for creating dynamic, responsive user interfaces that can update in real time based on ROS data.

2. PHP & MariaDB

PHP is one of the most common web-programming languages. Using the language extensive functions list provides fast and flexible development. MariaDB is a cutting edge open source database, which pairs very nicely with PHP for the robot interface back-end.

3. Python (Flask, Django)

Python is often used for backend development, where a web server running Flask or Django can serve as the intermediary between the web interface and the ROS environment. Using Python libraries like roslibpy, developers can handle ROS communication on the server side, process data, and serve it to the web client.

D. Advantages of Web Interfaces for ROS Robots

1. Remote Access and Control

Web interfaces enable users to control and monitor ROS robots from any location, as long as there is internet connectivity. This is particularly useful for applications where robots operate in remote or hazardous environments, such as search-and-rescue missions, industrial monitoring, or agricultural automation.

The ability to remotely update configurations, monitor real-time sensor data, and troubleshoot problems without physical access to the robot reduces downtime and enhances operational efficiency.

2. Scalability

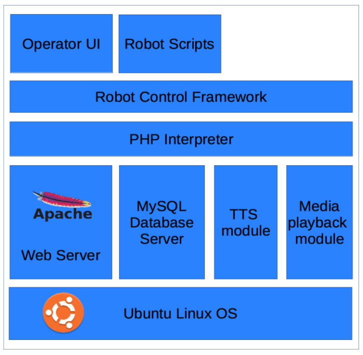
Web interfaces can be designed to manage multiple robots simultaneously, making them ideal for fleet management. Users can monitor the status of multiple robots, issue commands, and coordinate tasks through a single interface, streamlining operations across large deployments.

By leveraging cloud services, web interfaces can scale to accommodate more users and devices, allowing for centralized management and easier collaboration.

3. Cross-Platform Compatibility

Web interfaces are platform-independent, meaning they can be accessed from desktops, tablets, and smartphones. This flexibility allows users to interact with ROS robots from different devices, ensuring a consistent user experience regardless of the platform.

Figure 1 shows an IR BAS implementation of an web-based interface for ROS educational robots (BeBot & MaxiBot)



1. Figure 1: Embedded Web Interface for Educational Robot

E. Challenges of Web Interfaces in ROS

- Network Latency and Reliability

Effective communication between the web interface and ROS nodes relies on stable, low-latency network connections. High latency can result in delayed responses, which can be critical for real-time control tasks. Ensuring network reliability is essential, especially in scenarios where the robot needs to respond quickly to user commands.

In environments with unreliable internet connections, local networks or hybrid edge-cloud architectures may be used to mitigate latency issues.

- Security

Since web interfaces can be accessed remotely, security is a major concern. Robust authentication, encryption, and data protection protocols must be implemented to prevent unauthorized access to the robot's control system. Vulnerabilities in the web application can be exploited to interfere with robot operations, leading to potential safety risks.

Secure practices, including the use of SSL certificates, firewalls, and regular security audits, are necessary to protect sensitive data and control channels.

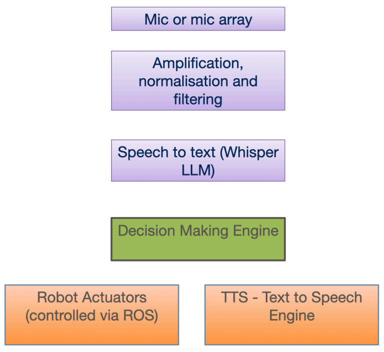
- Complexity of Integration

Integrating web interfaces with ROS requires a solid understanding of both web development and ROS communication protocols. Developers must handle multiple technologies, including WebSockets, REST APIs, and ROS messaging, to create smooth and responsive interfaces.

Developing a robust system that can handle real-time data while maintaining reliable operation across different network conditions can be relatively complex and time-consuming.

III. Voice Interface implementation

A voice interface for service robots involves a streamlined process of capturing, processing, and interpreting voice commands to allow seamless interaction. The system integrates hardware components (microphones, preamps) with software modules (speech recognition, diarization, decision-making) to achieve efficient and accurate voice control - block diagram shown on fig.2.


Mic or mic array
Amplification, normalisation and filtering
Speech to text (Whisper LLM)
Decision Making Engine
Robot Actuators (controlled via ROS)
TTS - Text to Speech Engine

2. Figure 2: Reference Voice Interface Implementation (IR-BAS)

A. Audio Capture: Microphone and Preamp

The process starts with capturing the user's voice through microphones, which can be directional, omnidirectional, or arrays depending on the environment. Microphone arrays are particularly effective as they enable features like beamforming to isolate the speaker's voice.

A preamplifier boosts the audio signal to ensure clarity and minimize noise, preparing it for further processing.

B. Audio Processing: Normalization and Noise Reduction

The captured audio undergoes normalization to standardize volume levels and noise reduction to filter out background sounds. These steps are crucial for ensuring clear, consistent input for the next stages. Techniques such as spectral subtraction and adaptive filtering can help improve audio quality, especially in noisy environment. Our present setup does not include this, but we plan to experiment it in a next stage.

C. Speaker Diarization (optional)

Diarization identifies and separates different speakers within the audio input, ensuring that commands intended for the robot are accurately detected. This is vital in multi-user scenarios. Machine learning models analyze voice characteristics to segment and attribute speech, helping the robot focus on relevant commands. It can be done using an external tool such as pyannote or NeMo.

Pyannote is an open-source toolkit designed for speaker diarization, or identifying "who spoke when" in audio recordings. Built on deep learning frameworks, pyannote provides highly accurate speaker segmentation and labeling, making it useful for applications in transcription, meeting analysis, and multi-speaker audio processing. It integrates seamlessly with audio processing workflows, offering pre-trained models and tools to fine-tune diarization for specific datasets.

NVIDIA NeMo Framework is a scalable and cloud-native generative AI framework built for researchers and PyTorch developers working on Large Language Models (LLMs), Multimodal Models (MMs), Automatic Speech Recognition (ASR), Text to Speech (TTS), and Computer Vision (CV) domains. It is designed to help you efficiently create, customize, and deploy new generative AI models by leveraging existing code and pre-trained model checkpoints. It provides a modular framework and pre-trained models, enabling researchers and developers to customize AI applications with minimal effort.

D. Speech-to-Text Conversion with Whisper LLM

The processed audio is then passed to a speech-to-text (STT) engine, such as Whisper, a large language model (LLM) that is very good in converting spoken language into text. Whisper's handles multiple languages, accents, and noisy conditions, making it optimal for diverse environments. We use implementation whisper-cpp. There reference tool **command**, which works with keyword commands, ready for interpretation. Also using the tool **stream** to get text into file and parsing in realtime with python is an working option.

E. Decision-Making Engine

The text output from the STT engine is processed by a decision-making engine. This component uses Natural Language Processing (NLP) to understand user intent, context, and execute appropriate actions. The engine may employ rule-based systems for straightforward commands and machine learning for more complex instructions. It translates voice inputs into robot commands, ensuring precise control and feedback when necessary.

Voice interfaces for service robots combine audio hardware with advanced software to enable accurate and intuitive control. By utilizing robust STT engines like Whisper and sophisticated decision-making systems, these interfaces can handle complex, real-world interactions. Continued advancements in noise reduction, diarization, and contextual NLP will further enhance the reliability and adaptability of voice-controlled robots, enabling smoother and more natural human-robot interactions.

VI. Conclusion

The development of a voice-controlled interface for ROS-based service robots in healthcare presents a transformative approach to enhancing interaction in high-stakes, hands-free environments.

By integrating speech recognition, natural language processing, and ROS middleware, these robots can interpret and execute verbal commands, allowing healthcare personnel to manage tasks with minimal physical interaction—an essential benefit in contaminated and infectious settings where reducing touchpoints can mitigate the spread of pathogens. This approach enables robots to assist with routine tasks, freeing healthcare workers to focus on patient-centered care and improving the overall efficiency of hospital operations.

However, implementing effective voice-controlled interfaces in healthcare brings significant technical challenges, particularly in achieving real-time responsiveness and low latency. In hospital settings, where immediate and accurate responses are critical, even slight delays in processing voice commands could hinder workflow.

To address this, edge computing is increasingly employed to perform processing tasks locally, significantly reducing latency and allowing the robot to respond promptly. Additionally, edge computing helps offload critical tasks from cloud-based systems, ensuring that the robot's functionality remains consistent even with fluctuating network connectivity—an important factor in real-time healthcare environments.

Ultimately, voice-controlled ROS service robots have the potential to revolutionize healthcare by enabling efficient, safe, and hygienic interaction. The application of advanced audio processing models like Whisper for speech-to-text conversion, combined with ROS's modular design, ensures that these robots can adapt to noisy, fast-paced healthcare settings while maintaining reliability. As advancements in voice recognition, real-time processing, and edge computing continue, voice-controlled healthcare robots will play a crucial role in reducing cross-contamination risks, improving task efficiency, and supporting medical personnel in critical ways. This technology is poised to significantly elevate standards of patient care, safety, and operational effectiveness in healthcare environments.

VII. ACKNOWLEDGMENT

The Authors Acknowledge The Financial Support Of The Project With Administrative Contract № KP-06-H57/8 From 16.11.2021. "Methodology For Determining The Functional Parameters Of A Mobile Collaborative Service Robot Assistant In Healthcare", Funded By The "Competition For Funding Basic Research - 2021." From The Research Sciences Fund, Bulgaria.

VIII. References

1. G. Angelov, Y. Paunski, R. Zahariev, N. Valchkova, "Designing a Web Based Software Control System for Service Robot and Sequence Programming Using Scripting Language Editor", Proceedings of the International Conference"Robotics & Mechatronics and Social Implementations 2018", CCS vol.1 08. 2018
2. Angelov, G., et al. (2011) "Remote Interface Communication to ROS Based Robotic System"; Proceedings of the Twenty First International Conference Robotics and Mechatronics; "Invited Session - Austrian-Bulgarian Automation Day"; 19-21 September 2011; Varna Bulgaria; ISSN 1310-3946; pp. 22–27.
3. https://github.com/openai/whisper
4. https://github.com/ggerganov/whisper.cpp
5. https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/starthere/intro.html
6. https://lajavaness.medium.com/comparing-state-of-the-art-speaker-diarization-frameworks-pyannote-vs-nemo-31a191c6300
7. https://github.com/pyannote/pyannote-audio/blob/develop/FAQ.md
8. https://wiki.ros.org/noetic
9. https://help.ubuntu.com/20.04/ubuntu-help/index.html