Academic

Development of a Robotic Arm Interface with Image and Voice Database Integration

Project Overview

Collaborated with Hiwin Technologies to develop a real-time, intuitive robotic arm interface integrating both image and voice commands. The system combined PLC and HMI controls to simulate various electronic components, sensors, and motors, enabling multi-behavior task execution.

My Contributions

- Designed and built image and voice databases, and developed matching algorithms for command execution.
- Integrated gesture and voice recognition technologies to enable seamless multimodal interaction.
- Developed a client-server architecture separating recognition and motion control tasks.
- Implemented automation safety mechanisms to enhance the safety of human-robot interaction.

Key Features

Voice Recognition

Real-time speech-to-text processing that maps verbal commands to predefined robotic actions, enabling natural language interaction.

Gesture Recognition

Real-time hand gesture recognition using MediaPipe.

Applied scaling and transformation matrices to accurately map image coordinates to robotic workspace coordinates.

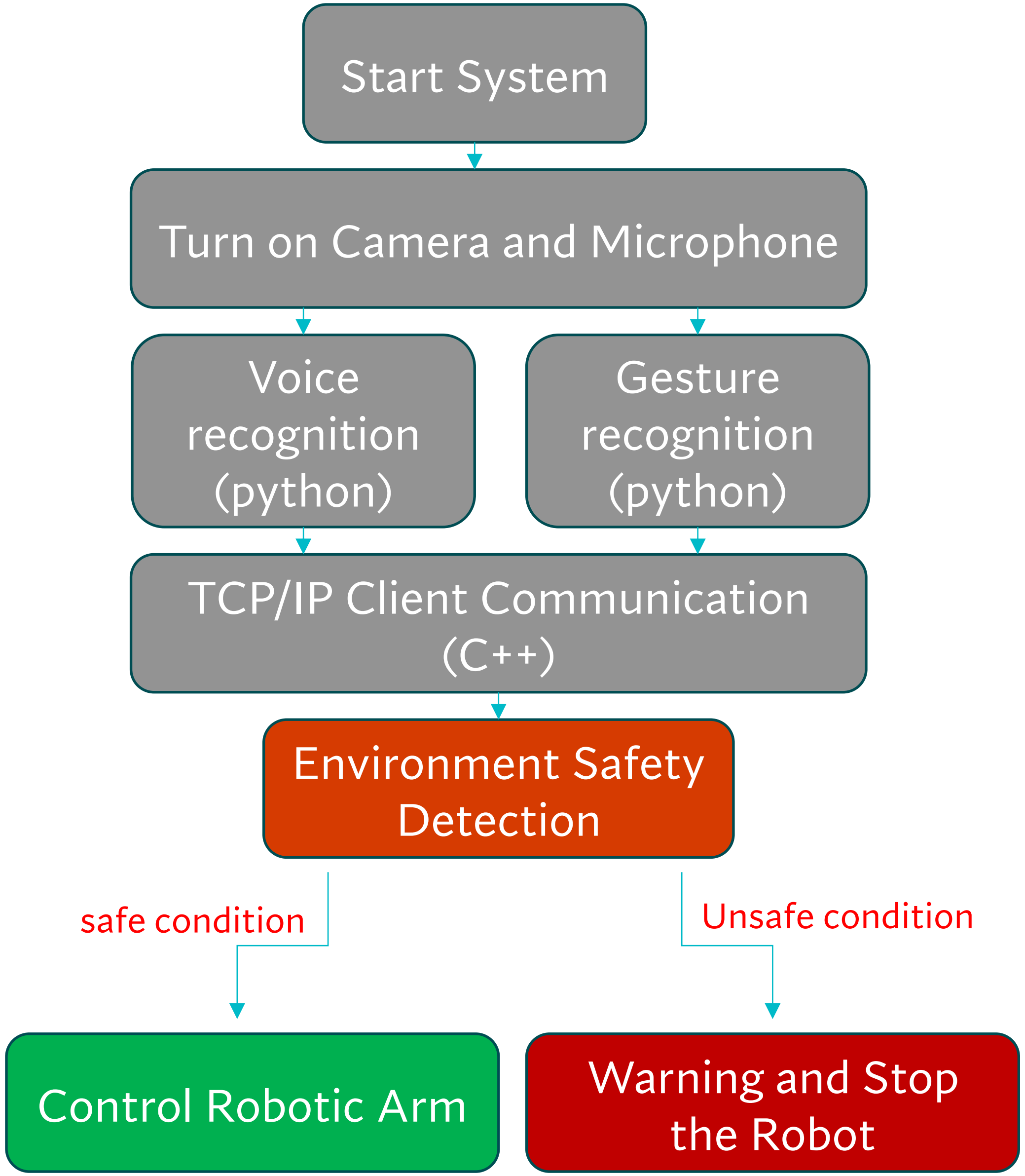
Client-Server Architecture

Python (Client): Handled voice and gesture recognition, translating inputs into control commands and coordinate data.

C++ (Server): Received commands via non-blocking TCP/IP sockets, executed inverse kinematics and motion interpolation, and controlled robotic movements through a robot control library.

Automation Safety Mechanism

Integrated proximity sensing and motion monitoring into the PLC system, implementing a collaborative safety alarm system that triggers real-time warnings or emergency stops to ensure safe human-robot interaction.


Start System
Turn on Camera and Microphone
Voice recognition (python)
Gesture recognition (python)
TCP/IP Client Communication (C++)
Environment Safety Detection
safe condition
Unsafe condition
Control Robotic Arm
Warning and Stop the Robot

Detailed System Integration Flow

1.Start System

- •Initialize all modules and hardware.

2.Turn On Microphone and Camera

- •Activate input devices:
  - •Microphone for voice capture.
  - •Camera for image capture.

3.Voice Recognition (Python)

- •Capture live audio stream.
- •Apply Speech-to-Text (STT) processing.
- •Match recognized command with database.
- •Generate corresponding robotic arm control command.

4.Gesture Recognition (Python)

- •Capture real-time video frames.
- •Detect hand landmarks using MediaPipe.
- •Analyze hand gestures.
- •Apply scaling and transformation matrices to map 2D camera coordinates to 3D robot coordinates.
- •Generate corresponding robotic arm control command.

5.TCP/IP Client Communication

- •Package voice and gesture data into structured command format.
- •Establish TCP/IP connection with server.
- •Renew and upload command and coordinate data in real-time (non-blocking socket).

6.Server Processing (C++)

- •Receive and parse incoming command data.
- •Validate command structure and content.
- •Calculate robotic arm trajectory using Inverse Kinematics (IK).
- •Perform Motion Interpolation for smooth movement planning.

7.Environment Safety Detection

- •Continuously read proximity sensor data via PLC.
- •Analyze environment for obstacles or unsafe conditions.
- •Predict potential collision based on current trajectory and environment mapping.

8.Decision Making

- **•If Safe Condition Detected:**
  - •Approve and execute movement command.
  - •Send control signals to robotic arm via motion control library.
- **•If Unsafe Condition Detected:**
  - •Trigger safety alarm.
  - •Immediately stop robotic arm movement (Emergency Stop).
  - •Notify user with warning signal (visual or audio).

9.Control Robotic Arm

- •Send real-time commands to servo motors.
- •Monitor arm position and movement feedback.

10.System Feedback and Monitoring

- •Display system status on HMI (e.g., Running, Error, Stopped).
- •Log all operations and safety events for record-keeping.