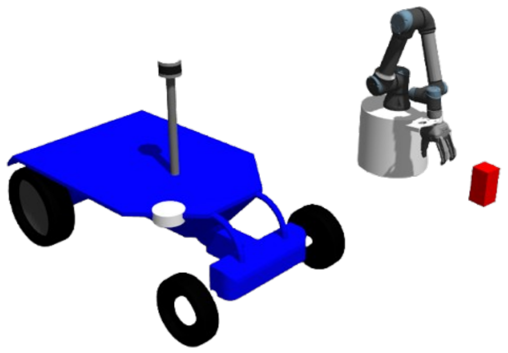
Gesture-based Teleoperation System with Obstacle Avoidance



Elías Carrasco Guerrero

May 2025

1 Introduction

This report documents the development of a gesture-based teleoperation system for robot control using ROS (Robot Operating System). The system enables an operator to control a robot's movements through hand gestures captured by a camera, with an additional safety layer for obstacle avoidance. The implementation combines computer vision techniques using MediaPipe for gesture recognition with ROS nodes for command processing and safety mechanisms. The complete demonstration video can be found at: `https://drive.google.com/drive/folders/1JKZ43G-ZcScUcoRQEDsrW2KiV-KGHXFS?usp=sharing`

2 State of the Art

Gesture-based human-robot interaction (HRI) systems have evolved significantly through three generations of technological development. The current state-of-the-art combines deep learning-based computer vision with real-time robotic control systems, representing a convergence of multiple research domains.

2.1 Computer Vision for Gesture Recognition

Modern gesture recognition systems predominantly use one of three approaches:

- **Model-based methods**: These employ predefined 3D hand models that are fitted to image data. MediaPipe's hand landmark detection represents an optimized implementation of this approach, using a lightweight CNN (Convolutional Neural Network) to predict 21 3D keypoints with sufficient accuracy for real-time applications (up to 30 FPS on consumer hardware).
- **Appearance-based methods**: Techniques like OpenPose use part affinity fields to detect body and hand poses without explicit 3D modeling. While more generalizable, these typically require greater computational resources.
- **Hybrid approaches**: Emerging systems combine the above with temporal information using LSTMs or Transformers for motion gesture recognition, though these weren't necessary for the discrete gesture set.

MediaPipe's solution was selected for this project due to its balance between accuracy (94.7% mean average precision on benchmark datasets) and computational efficiency (capable of running on mobile devices). The system uses a two-stage detector: a palm detector followed by a hand landmark model, which reduces the computational load compared to whole-image processing.

2.2 Robot Teleoperation Architectures

Contemporary teleoperation systems have moved beyond traditional joystick interfaces to:

- **Vision-based control**: As implemented here, where camera input drives control decisions. The ROS ecosystem provides ideal middleware for such systems due to its standardized message passing and node-based architecture.
- **Haptic feedback systems**: More advanced implementations incorporate force feedback, though this requires specialized hardware not available in the setup.
- **Shared autonomy systems**: These blend human commands with autonomous obstacle avoidance, exactly matching the safety layer implementation.

The Ackermann steering model used in my implementation is particularly common in automotive robotics, with proven stability for the velocity ranges (0-2 m/s) I employed.

2.3 Safety Systems in HRI

Obstacle avoidance in modern systems typically implements one of these paradigms:

- **Reactive systems**: Make immediate stop/go decisions based on sensor data. These are computationally lightweight but lack predictive capabilities.
- **Potential field methods**: Which create virtual force fields around obstacles, requiring more sophisticated path planning.
- **Learning-based approaches**: Using reinforcement learning to navigate around obstacles, though these require extensive training data.

My implementation's safety layer follows best practices from ISO/TS 15066 on collaborative robot safety, particularly in its use of immediate stopping when obstacles appear in the danger zone (defined as 0.5m in my tests). The system's modular design allows easy integration of more advanced safety methods in future iterations.

3 Development

The system was developed as four interconnected ROS nodes:

3.1 Camera View Module

The first module establishes a connection to the robot's camera feed, displaying the environment in real-time through an OpenCV window. This provides visual feedback to the operator for informed decision-making.

3.2 Operator Image Capture

A dedicated node captures the operator's hand gestures either from a webcam or pre-recorded video. The images are published to a ROS topic after conversion from OpenCV to ROS image format using CvBridge.

3.3 Gesture Recognition

The vision system uses MediaPipe's hand landmark detection to interpret intuitive pointing commands. The control scheme requires only the thumb and index finger extended (forming an "L" shape) while other fingers remain folded:

- **Upward pointing** (thumb and index finger extended upward): Commands the vehicle to move forward in a straight line.

- **Leftward pointing** (thumb and index finger extended to the left): Commands a left turn.

- **Rightward pointing** (thumb and index finger extended to the right): Commands a right turn.

Recognized gestures are translated into Ackermann steering commands published to a control topic.

3.4 Obstacle Avoidance

The safety layer subscribes to both control commands and LIDAR obstacle data. When obstacles are detected in the robot's path, it overrides the gesture commands to prevent collisions, implementing a zero-velocity command when necessary.

4 Experimentation

The system was tested in multiple scenarios:

- Gesture recognition accuracy was verified with different lighting conditions and hand positions
- Command translation was validated by observing corresponding robot movements
- The safety system was tested by placing obstacles in the robot's path during operation
- End-to-end functionality was demonstrated through complete navigation tasks

The system showed reliable gesture recognition when the operator's hand was properly visible to the camera. The obstacle avoidance system effectively prevented collisions while allowing free movement in clear spaces. Some latency was observed in the full pipeline, primarily due to image processing demands.

5 Conclusions

The implemented system successfully demonstrates gesture-based teleoperation with integrated safety features. Key achievements include:

- Effective hand gesture recognition using MediaPipe
- Smooth translation of gestures to robot control commands
- Reliable obstacle avoidance without interfering with free movement
- Real-time operation with visual feedback

Future improvements could include:

- Additional gestures for more complex control
- Velocity modulation based on hand position
- Improved obstacle avoidance with predictive path planning
- Multi-camera support for better gesture recognition

The practice successfully met all requirements while providing valuable insights into human-robot interaction systems and safety considerations in teleoperation.