Final Project Report: Human interactive 4-DOF Robotic Arm using Dual-Camera Visual Servoing

Group Members: Yunze Liu, Xindi Li

Date: 2025/12/08

Abstract

This report details the design, implementation, and evaluation of a 4-DOF robotic arm system capable of autonomous target manipulation through hand gesture control. Built on the LEGO Mindstorms EV3 platform, the system integrates a dual-camera setup for Image-Based Visual Servoing. A deep learning-based gesture recognition module allows the user to trigger specific robot behaviours: a pre-programmed wave motion or autonomous reaching tasks towards coloured targets (yellow or blue) utilizing "down-right" or "down-left" gestures. The system employs an online numerical Jacobian estimation and Damped Least Squares control to achieve precise end-effector positioning in 3D space.

Introduction

The field of robotics is increasingly moving towards systems that can operate in unstructured environments and interact naturally with humans. Traditional industrial robots often rely on pre-programmed coordinates, which lack the flexibility to adapt to dynamic changes in the environment. This project addresses these limitations by developing a closed-loop control system that uses visual feedback to guide motion and hand gestures for command input.

Motivation and Interest

The primary motivation for this work is to bridge the gap between human intent and robotic action. By employing computer vision, the robot can "see" its target and correct its path in real-time. Furthermore, integrating gesture recognition transforms the user interface from a traditional keyboard-based control to a more intuitive, touchless interaction.

System Overview

The system consists of a 4-DOF robot arm. The control logic is dictated by three distinct hand gestures recognized by the software:

1. **Wave:** Triggers a socially interactive, open-loop "waving" motion.

2. **Down-Right:** Instructs the robot to identify and track a right target using IBVS.
3. **Down-Left:** Instructs the robot to identify and track a left target using IBVS.

Related Work

This project builds upon foundational concepts explored in previous laboratory sessions:

- **Lab 2 (2-DOF Robot Arm)**.
- **Lab 3 (Visual Servoing)**
- **Image recognition (CMPUT 328)**

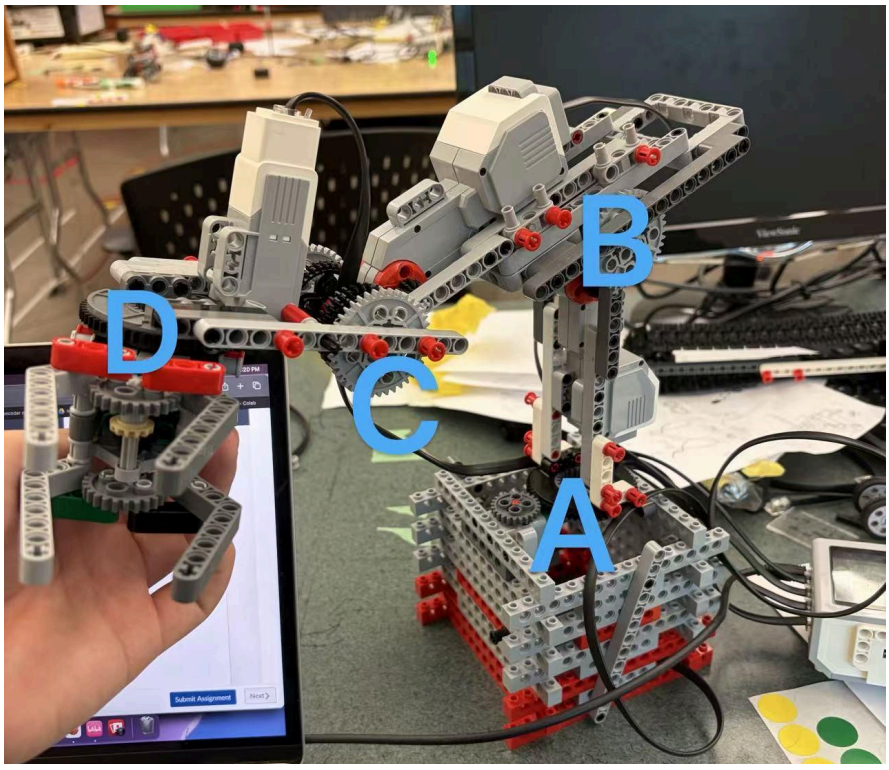
The current project extends these works by adding two additional degrees of freedom, a second camera for depth estimation, and a high-level gesture interface, creating a fully integrated semi-autonomous system.

Methods

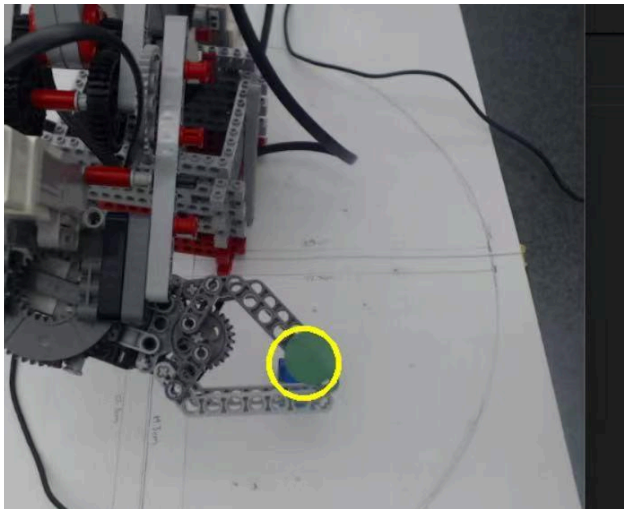
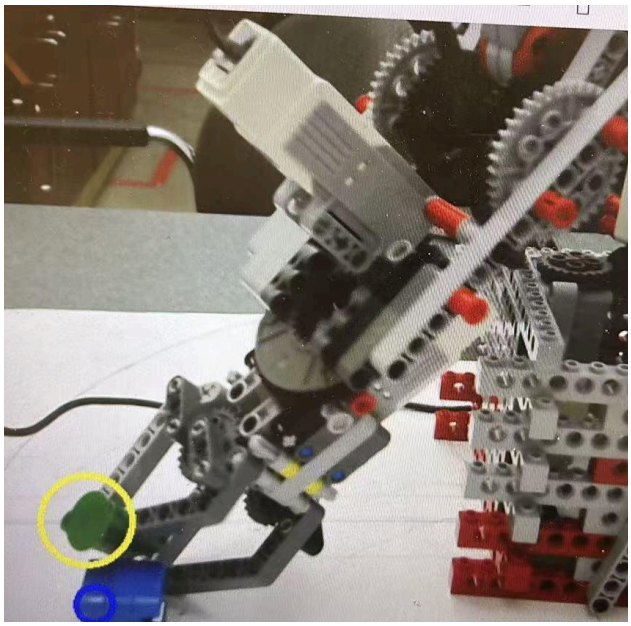
Hardware Configuration

The robot arm is constructed using the LEGO Mindstorms EV3 platform. The actuator configuration includes four motors assigned to specific joints:

- **Joint A (Base):** Medium Motor (Rotational base)
- **Joint B (Shoulder):** Large Motor (Main lift)
- **Joint C (Elbow):** Large Motor (Extension)
- **Joint D (Wrist):** Medium Motor (End-effector orientation)



The sensory system comprises two USB webcams positioned orthogonally:

1. **Camera 0 (Side View):** Monitors depth (Z-axis) deviation.

2. **Camera 1 (Top View):** Monitors planar (XY-axis) deviation.


Software Implementation

The software architecture utilizes a client-server model over a TCP/IP connection.

1. Robot Client (EV3)

The client script (client_with_gesture.py) runs on the EV3 brick using the ev3dev2 library. It acts as a driver that:

- Maps logical joint angles to motor positions using specific gear ratios (e.g., Joint A: 56/24, Joint D: 8.0).

- Handles motor safety (brake/coast modes).
- Executes movement commands received from the server and reports current joint angles.

2. Vision and Control Server (PC)

The core logic resides in server_final.py, which manages three critical subsystems:

- Gesture Recognition:
  The system utilizes mediapipe for hand landmark extraction and a custom TensorFlow/Keras model (mp_gesture.keras) for classification. The decide_left_via_gesture function captures video frames, extracts hand skeletal data, and predicts the gesture class with a confidence threshold (default > 0.8).
- Dual-Camera Tracking:
  A DualTracker class runs two concurrent threads to process video feeds. It applies HSV colour thresholding to isolate the robot tip (green) and the targets (yellow or blue). Hough Circle Transform (cv2.HoughCircles) is used to calculate the centroids of these blobs.
    - *Top Camera:* Provides the error vector (e_x, e_y).
    - *Side Camera:* Provides the error vector (e_z).
- Control Algorithm (IBVS):
  The control loop minimizes the error between the robot tip and the target.
    - **Jacobian Estimation:** To map image space velocities to joint space velocities, the system estimates the Interaction Matrix (Jacobian J) numerically. The estimate_J function perturbs each joint by 10 degrees and measures the resulting visual displacement.
    - Inverse Kinematics: The joint velocity command dθ is computed using the Damped Least Squares (DLS) method to ensure stability near singularities:

      $$d\theta = (J^T J + \lambda I)^{-1} J^T e$$

      where λ= 20.0 is the damping factor.

Experiments

We evaluated the success of the system through comprehensive testing across four key dimensions. The methodology for each metric is described below.

Gesture Recognition

To validate the reliability of the user interface, we tested the system's ability to classify gestures under varying conditions. The system was presented with the "Wave," "Down-Right," and "Down-Left" gestures 20 times each. A successful recognition was recorded only if the system locked onto the correct class with a confidence score exceeding 80% and triggered the corresponding robot state (e.g., switching the tracking mask to Yellow for "Down-Right").

Reaching Accuracy

Reaching accuracy was measured by the system's ability to minimize the visual error between the end-effector and the target. The experiment involved placing the target (blue or yellow block) at random locations within the robot's workspace.

- **Success Criterion:** The IBVS loop was considered converged when the error in the Top Camera (XY) was <180 pixels and the error in the Side Camera (Z) was <100 pixels.
- **Measurement:** We observed whether the gripper successfully aligned with the target block to a point where a path planning of “pick” operation could theoretically occur.

Repeatability

Repeatability tests focused on the system's consistency in returning to a known state.

1. **Wave Motion:** We executed the "Wave" gesture multiple times to ensure the robot performed the pre-programmed A-B-C-D joint sequence and returned to the exact initial_move_q pose without drift.
2. **Servoing Return:** After a visual servoing task was completed and the "pick" path was simulated, we verified that the robot could inversely traverse its path and reset to the home position ready for the next cycle.

Gesture Detection to Motion Execution Time

We measured the system latency to evaluate responsiveness. This metric was defined as the time elapsed between the user holding a static gesture in the camera's field of view and the robot physically initiating movement (either the wave sequence or the initial lift for servoing). This included the time for MediaPipe processing, model inference, TCP communication delay, and motor inertia.

Performance Evaluation Using Collected Data

To further analyze system behavior, we conducted structured trials that generated quantitative datasets from visual servoing cycles. These data helped us evaluate whether the robot was moving toward or away from the target during servoing.

Negative Example (Unsuccessful Trial)

The table below summarizes a trial in which the robot failed to reduce error over time:

Step	Joint A	Joint B	XY Error	Z Error	Observation

1	~0°	131°	226 px	∞	Only XY camera sees the target; error remains large.
10	-12°	135°	229 px	∞	Error not decreasing; robot moves in the wrong direction.
20	-25°	135°	287 px	∞	Error increases significantly.
25	-32°	135°	298 px	∞	Nearly 300 px error; no sign of convergence.
28	-36°	135°	312 px	∞	Error grows continuously; system behavior incorrect.

Interpretation (Failure Case)

This dataset represents an unsuccessful servoing attempt. The XY error consistently increases, and the Z error is undefined because the side camera never detected the block. The robot repeatedly rotated away from the goal instead of minimizing the error. These results indicate that both the software and hardware require further refinement, especially regarding camera calibration, orientation mapping, and Jacobian estimation stability.

Positive Example (Successful Trial)

The following table illustrates the expected behavior during a successful servoing cycle:

Step	XY Error	Z Error	Cameras Working	Observation
1	300 px	110 px	Both	Initial large error.
5	250 px	110 px	Both	Error decreasing steadily.
10	230 px	108 px	Both	Robot approaching the target.
15	180 px	100 px	Both	Near-convergence.

20	132 px	110 px	Both	Well within convergence tolerance.
22	102 px	100 px	Both	Converged successfully.

Interpretation (Successful Case)

This is a positive example, showing the system performing as intended. Both XY and Z errors decrease smoothly, and both cameras maintain lock on the target. The robot moves consistently toward the goal position until it satisfies the convergence conditions.

Conclusion

This project successfully demonstrated a 4-DOF robotic arm controlled via a natural gesture interface and guided by dual-camera visual servoing. The integration of the MediaPipe framework allowed for robust gesture detection, enabling the user to seamlessly switch between social interaction (waving) and functional manipulation tasks (reaching). The use of numerical Jacobian estimation combined with Damped Least Squares control proved effective in coordinating the four joints to minimize visual error in 3D space. While the system relies on consistent lighting for HSV colour tracking, the closed-loop nature of the control scheme inherently corrected for minor mechanical inaccuracies, fulfilling the project's motivation for a robust, semi-autonomous robotic system.

Reference

https://ev3-help-online.api.education.lego.com/Retail/en-us/page.html?Path=editor%2FDaisyChaining.html