Project Proposal: Multi-Target Visual Servoing with Residual Learning

Qihao Qian

UCSD ECE Department
A69030355
q2qian@ucsd.edu

I. Project Objective

A. Robot Task

In a planar environment with multiple static targets randomly placed, the robot equipped with an eye-in-hand RGB-D camera autonomously selects the target closest to its base and controls the camera through visual servoing to drive the target toward the center of the captured image.

B. Research Focus

Compare traditional **Image-Based Visual Servoing (IBVS)** with an enhanced method **IBVS + Neural Residual Compensation** under multi-target scenarios. Evaluate how residual learning compensates for unknown model errors and local nonlinearities in terms of precision and convergence speed.

II. Baseline Method and Limitations

A. Baseline

The classical IBVS joint-space controller introduced in class:

$$\delta\mathbf{q} = K_p J^\dagger L_s^\dagger \begin{bmatrix} u_{\text{des}} - u \\ v_{\text{des}} - v \end{bmatrix} \quad (1)$$

where $\delta\mathbf{q}$ is the commanded joint increment, $J^\dagger$ is the robot Jacobian pseudo-inverse, and $L_s^\dagger$ is the image Jacobian pseudo-inverse mapping pixel errors into camera twists.

B. Limitations

- Approximation of the Jacobian and depth estimation errors cause significant residuals and inconsistent convergence under varying target poses and viewpoints.
- The control law lacks robustness against unmodeled dynamics and discrete sampling noise.
- In multi-target settings, frequent re-targeting amplifies accumulated errors, making it difficult to balance speed and stability with a fixed gain.

III. Proposed Method and Rationale

A. Alternative Approach

Residual Visual Servoing (RVS) based on neural residual learning.

B. Core Idea

- Input: single RGB frame and normalized pixel coordinates.
- A lightweight CNN predicts the **residual joint increment**

$$\Delta\mathbf{q} = f_\theta(I, \tilde{u}, \tilde{v}, \tilde{u}^*, \tilde{v}^*) \quad (2)$$

- Online control law:

$$\delta\mathbf{q} = \delta\mathbf{q}_{\text{IBVS}} + \Delta\mathbf{q} \quad (3)$$

  where the residual is bounded using `tanh` and joint-limit-aware clipping (implemented, e.g., via `clamp`) to maintain task stability.

C. Data Collection

- Run baseline IBVS joint-space control in PyBullet.
- Record image frames, pixel errors, IBVS joint commands, and the "teacher" joint deltas derived from geometric ground truth as supervision targets (residuals).

D. Expected Outcomes

- Compensate for depth estimation bias, local nonlinearities, and environmental disturbances.
- Improve convergence speed and maintain consistent tracking across random multi-target scenes.
- Use CNN-based visual context to infer implicit errors from object scale or illumination variation.

IV. Simulation Platform and Setup

A. Simulator

PyBullet (GUI mode)

B. Robot

Franka Emika Panda (official URDF, with wrist-mounted eye-in-hand camera)

C. Environment Setup

- `create_physics_environment`: randomly generates n static cube targets.
- At each frame, select the target nearest to the robot base.
- Scripts:
  - `collect_residual_data.py` — residual dataset collection.

- `train_residual_net.py` — neural network training.
- `simulate_residual_control.py` — online control validation.

D. Implementation Notes

- Use standardized 512×512 RGB images.
- Small CNN backbone with Batch Normalization (BatchNorm).
- PyTorch for deep learning; NumPy / OpenCV / PyBullet for simulation and data processing.

E. Comparison Experiments

- Execute both **pure IBVS** and **Residual VS** under identical target sets.
- Evaluate:
  - Pixel error trajectories
  - Convergence time
  - Control energy, e.g.

$$\int \|\delta \mathbf{q}(t)\|^2 \, \mathrm{d}t \tag{4}$$