Peg Insertion Simulator - Personal Project Technical Report

Author: Sayan Mondal

Date: Aug 11, 2025

Abstract

This personal project builds a realistic peg-in-hole manipulation simulator in MuJoCo using a Robotiq parallel gripper (sourced from the "mujoco_menagerie") and two provided assets: a single peg STL and a single cuboid-with-hole STL. I engineered a 6-DoF controllable base for the gripper (3T + 3R) and a 1-DoF jaw actuator, supporting both teleoperation and agent-based control. The system emphasizes faithful contact dynamics (stable insertion without interpenetration/explosive forces), a clean control API (6D pose + jaw separation), and re-implementability. This report documents modeling decisions, physics parameters, controller design, software structure, and a step-by-step recipe to reproduce the setup.

Code availability

The implementation is not publicly hosted. For access to the code, please contact the author (Sayan Mondal).

1. Problem & Goals

Objective. Build a physics-realistic peg insertion environment with:

- A 6-DoF controllable end-effector (EE) and a parallel-jaw gripper.
- Repeatable contact dynamics enabling reliable alignment and insertion.
- Two operation modes: (i) teleoperation, (ii) simple scripted/agent control.
- Clean abstractions so others can re-implement from the write-up alone.

Constraints. Only one peg STL and one cuboid-hole STL were provided; the gripper model was imported from "mujoco_menagerie". No mocap bodies; control is via joints/actuators.

Success Criteria. Insertion without jamming or tunneling, with bounded contact forces, and recoverable failure modes (e.g., re-alignment after light mis-placement).

2. System Overview

Key components.

1) Assets: peg mesh, cuboid-hole mesh, Robotiq gripper.

2) Modeling: 6-DoF base joint stack for the gripper + 1-DoF jaw.

3) Control: 6D pose command -> low-level controller -> MuJoCo actuators.

4) Modes: teleop input mapping; scripted/agent policy.

5) Evaluation: success checks, force/pose logging, repeatable seeds.

High-level data flow.

```
User/Agent Command (x, y, z, roll, pitch, yaw, jaw)
        |
Task-Space Controller (pose error -> twist/effort)
        |
Joint/Actuator Commands (base 6-DoF, jaw 1-DoF)
        |
MuJoCo Physics (contacts, friction, compliance)
        |
State/Logs (EE pose, forces, success/failure)
```

Note: The controller avoids mocap. The base pose is realized with a stack of MuJoCo joints/actuators (see Section 4), enabling both teleop and agent control through the same API.

2.1 Coordinate frames & notation

- World frame (W): right-handed; Z up.

- Gripper/EE frame (G): origin at jaw midline; +X forward, +Z up in open pose.

- Cuboid frame (C): centered in the block; rotation identity unless randomized.

- Units: meters, radians, seconds.

- Quaternions: [w, x, y, z]; body-fixed angular velocity omega.

3. Assets & Modeling

3.1 Robotiq Gripper (Import)

- Source: "mujoco_menagerie" Robotiq parallel gripper (open-source).

- Modifications: Added a controllable 6-DoF base and a jaw actuator for separation control.

3.2 6-DoF Base Joint Stack

- Translation (3-DoF): gripper_transX, gripper_transY, gripper_transZ - prismatic slides along X/Y/Z, range -10..10 m, damping 100.

- Rotation (3-DoF): gripper_rot - ball joint (replacing the earlier hinge stack to avoid gimbal-lock), damping 50, stiffness 0, armature 0.1.

- Base inertial: at the gripper_root body, mass 0.1 kg, diaginertia [0.01, 0.01, 0.01].

Why ball > Euler hinges? The single ball joint provides a minimal-singularity orientation parameterization and works cleanly with three axis-aligned torque motors (see Section 4.2), while the commented Euler hinges suffered from singularities around +-90 deg pitch.

4. Control & Interfaces

4.1 Command API

A single command vector drives everything:

u = [x, y, z, roll, pitch, yaw, jaw]

where the first six entries specify the desired EE pose (w.r.t. world or a task frame), and jaw is target separation.

4.2 Task-Space Controller

Goal. Track a desired pose (p_d, q_d) and jaw opening (s_d).

Notation. p in R^3 (position), v = p_dot (linear velocity), q (unit quaternion), omega (body-fixed angular velocity), tau (ball-joint torques).

Position control (slides X/Y/Z). Set absolute targets on gripper_transX/Y/Z using PID:

```
u_xyz = K_p * (p_d - p) + K_d * (v_d - v)
```

The command is bounded (per-axis) before being sent to the actuators.

Orientation control (ball joint). Compute a small-angle error from the quaternion difference and track a desired body-fixed angular velocity:

```
q_e = q_d (otimes) q^{-1}
theta = Log(q_e)  in R^3
omega_d = K_R * theta
tau = K_omega * (omega_d - omega)
```

Apply element-wise clipping: |tau|_inf <= tau_max.

Implementation note. Log(q) is the quaternion logarithm (minimal-angle axis-angle vector). For small angles: theta approx 2 * sign(q_e.w) * q_e.xyz.

Jaw actuation (tendon). A scalar command u in [0, 255] maps to tendon force:

```
F_tendon = g * u + b
```

Software converts u -> s (jaw separation, meters) via calibration.

Safety & filtering.

- Saturate all commands; low-pass filter omega.
- Apply soft velocity caps before sending to actuators.

4.3 Teleoperation

Keymap (chosen to avoid viewer conflicts):

+Y/-Y: 4 / 5

+X/-X: 6 / 7

+Z/-Z: 8 / 9

Roll: Q / E

Pitch: Up / Down

Yaw: Left / Right

Jaw open/close: [/]

Reset attempt: Z

Quit: ESC

Behavior. Keys generate small pose/jaw deltas each step. Rate limiting and deadzones prevent oscillation. Reset restores qpos, zeroes qvel, optionally re-randomizes objects, then runs one mj_forward.

4.4 Scripted/Agent Control

Phases (FSM):

1) Move above peg -> XY align to peg.

2) Descend to safe_descent_height.

3) Grasp: close to grasp_signal, wait grasp_time.

4) Lift to target_lift.

5) Rotate ~95 deg about X (track roll via body quaternion -> Euler, stop when |Delta theta| < rotation_error or timeout).

6) Align to cuboid in XY while holding height: compute gripper target XY = cuboid XY - (peg - gripper) XY offset.

7) Release: lower to safe release height, open fingers.

8) Settle/Complete: wait for drop or timeout, then COMPLETED.

Control law: incremental position steps of move_speed toward a target; angular velocity set to +-angular_velocity during rotation; jaw moves in small command steps each tick.

4.5 Runtime speed control

Base physics step 0.002 s with live multipliers 0.1x..5x (presets: key 4->0.5x, 5->1.0x, 6->2.0x).

4.6 Jaw command <-> separation calibration

- Control domain: actuator accepts a byte 0-255; tendon force is mapped by a linear gain and bias inside the XML. The jaw separation is empirically calibrated against this control to achieve 0-85 mm of opening.

- Practical note: the mapping is mildly nonlinear due to linkage geometry and contact; a piecewise-linear fit over the open/close trajectory is used in software to report the commanded separation and to decide grasp completion.

5. Physics & Simulation Setup

Defaults. timestep = 0.001 s, integrator = RK4, solver = CG, noslip_iterations = 20.

Global contacts. o_solref = 0.005 0.9, o_solimp = 0.9 0.9 0.01, cone = elliptic, impratio = 10.

Overrides.

- Peg: friction = [0.8, 0.1, 1e-4], solref = 0.005 1, solimp = 0.9 0.95 0.001, density 500; start approx (-0.2, 0.0, 0.015), 90 deg about X.

- Cuboid: 10 convex parts (object_01..10.obj), margin = 0.001, friction = [0.9, 0.1, 1e-4], solref = 0.01 1, solimp = 0.9 0.95 0.001, density 1000; visual cuboid.stl non-colliding; fixed via free joint with range 0 0.

- Finger pads: friction approx 0.7/0.6; solref = 0.004 1, solimp = 0.95 0.99 0.001.

Damping/armature. Slides: 100; ball: damping 50, armature 0.1.

Collision health. Exclude internal gripper contacts; maintain 4-bar via equalities; drive jaw via fixed tendon "split" (byte 0-255 -> force with gain/bias; forcerange -5..5).

Units. All meshes in meters; millimeter sources scaled by 1e-3.

5.1 Design Rationale: Integrator & Solver

- Why RK4? Higher local accuracy and good stability at 1 ms steps; reduces contact jitter versus semi-implicit Euler, yet avoids the heavy cost/tuning burden of fully implicit integrators.

- Why CG? Better contact resolution and lower jitter than PGS under tight frictional constraints, while remaining real-time -- crucial when gains are high or multiple contacts stack.

5.2 Contacts & Unachievable Commands

- Non-penetration as a feature: The CG solver clamps infeasible motion; when the commanded gripper pose would interpenetrate, constraints generate reaction forces and motion halts along the blocked directions.

- Incremental tracking: Commands are applied in small increments (teleop-style), letting physics naturally gate progress at contacts.

- Contact logging: Inter-body contact forces are recorded each step (with a small magnitude threshold) and categorized by pair (gripper, peg, cuboid, table) for analysis.

6. Tasks, Metrics & Logging

- Initializations: randomized peg start pose within tolerance; fixed hole pose.

- Success: peg depth >= threshold and lateral/rotational error <= thresholds for N consecutive steps.

- Failures: timeouts, force spikes, excessive tilt, or loss of grasp.

- Logs: EE pose, peg/hole relative transform, contact forces/impulses, command history.

Teleop insertion detection: horizontal_tolerance 0.03 m; insertion height threshold = cuboid center + 0.04 m; "settled" if peg speed <= 0.05 m/s.

Tolerances (agent): rotation_error 0.1 rad (approx 6 deg); alignment_distance 0.005 m; jaw_position tolerance 10.0 (controller units); height_completion 0.01 m.

Timing (agent): grasp_time 0.5 s; lift_timeout 3.0 s; rotation_timeout 12.0 s; alignment_timeout 4.0 s; positioning_time 2.0 s; release_time 4.0 s; settling_timeout 7.0 s.

Episodes & logging: teleop: max_timesteps 50,000, log_interval 100, save_interval 1000, completion_check_interval 500, max_attempts 5. Agent: max_timesteps 20,000, log_interval 50.

6.1 Logging schema (JSON)

Per-timestep entries include: commanded position, local_angular_velocity, jaw_separation (m); actual gripper position, euler_angles, quaternion, jaw_separation (from pad distance); peg position/orientation; and filtered inter-body contact forces (force vector, magnitude, and contact position) for pairs in {gripper, peg, cuboid, table}. Episodes finalize with status, total timesteps, duration, and file path.

6.2 Real-time performance & logging

- Synchronous logging at 1 kHz can stall the physics/render loop. A producer-consumer queue decouples logging from simulation. Optional throttling (e.g., log only near contact/when inserting) further reduces load.

7. Engineering Challenges & Fixes (Practical Notes)

What went wrong and how I fixed it.

- Gimbal lock (Euler stack) -> ball joint + omega-tracking. Replaced XYZ hinges with a ball joint; controlled using body-fixed angular velocity with torque clipping.

- High-gain instability. Added damping (ball=50), torque caps, and 1-2 ms steps; bounded speed multipliers.

- Insertion jitter. Tuned peg/cuboid solref/solimp; used slightly asymmetric pad friction to break symmetry.

- Non-convex block. Switched to 10 convex parts; kept per-geom friction consistent.

- Keyboard I/O edge cases. Separate press/release handlers to avoid "stuck rotation".

- Racey resets. Used a request_reset flag; perform one mj_forward after applying initial state.

- Jaw calibration. Mapped 0-255 byte to separation up to 85 mm via empirical fit; used separation for grasp timing.

8. Reimplementation Guide (Step-by-Step)

1. Import Gripper: pull Robotiq model from "mujoco_menagerie".

2. Add Base Joints: 3 prismatic (XYZ) + 1 ball joint for rotation; add actuators.

3. Add Jaw Actuator: single scalar command for separation with limits.

4. Load Meshes: peg + hole; confirm metric scaling and frames.

5. Tune Physics: timestep, solref/solimp, friction; validate contact stability with drop tests.

6. Implement Controller: position slides with PID; ball joint with omega-tracking PD + torque limits; jaw byte 0-255.

7. Teleop Mapping: keys 4/5/6/7/8/9 (XYZ), Q/E, arrows (roll/pitch/yaw), [/] (jaw), Z reset.

8. Scripted Policy: FSM steps 1-8 above; use offset-based XY alignment and 95 deg roll.

9. Define Metrics: success criteria; add logging hooks (pose, forces, contacts).

10. Experiment: sweep tolerances/gains; record success/failure examples.

11. Run:

- Teleop: python simulate_peg_insertion.py --mode teleop [--randomize]

- Agent: python simulate_peg_insertion.py --mode autonomous [--randomize]

9. Limitations & Future Work

- Physics/contact realism: explore soft contacts for smoother force profiles during tight fits; consider anisotropic friction.

- Control: add force/torque sensing and impedance/admittance loops; adaptive grasp completion using force signatures.

- Infrastructure: implement asynchronous logging via queue and background writer; add headless mode and adaptive timestep for sweeps.

- Task extensions: domain randomization beyond pose (mass, friction, tolerances); vision-based alignment with simulated RGB-D; RL/IL policies to replace or augment the FSM; automated failure analysis for success metrics and diagnostics.