# #6_UR
# Reality Gap Modeling & RL Motion Optimization (simulated + physical)

## Context

A fundamental challenge in robotics is the reality gap. Simulators produce idealized behavior, but real robots exhibit tracking errors, vibration, thermal drift, and load-dependent deviations. At Universal Robots, understanding and bridging this gap is critical for delivering accurate and efficient cobots.

In URSim (the UR simulator), all "actual" values are zero, meaning targets are perfectly tracked. On a real UR10e however, the actual joint positions, TCP pose, and forces deviate from targets depending on speed, acceleration, payload, and joint configuration.

This case challenges students to characterize the reality gap, learn to predict it, and then use reinforcement learning to optimize robot movements for speed while minimizing vibrations. The core tradeoff is intuitive: faster movements cause more oscillation at the end, which wastes time settling. The RL agent must learn to find the sweet spot.

## Challenge

The challenge is structured in three layers:

1. Analyze the reality gap: Using RTDE recordings from a real UR10e (multiple runs with varied speed, acceleration, and blend settings), characterize the deviation between target and actual values. Identify where and why tracking error and vibration occur, for example which joints are affected, at what speeds, and under what loads.

2. Learn the gap: Train a model (neural network, regression, or similar) that predicts actual robot behavior given target commands and motion parameters: actuals = f(targets, joint_config, velocity, acceleration, payload). A linear regression baseline is provided. This learned model captures what URSim cannot: the real robot's imperfections. Because the data covers varied parameter settings, the model can predict how changing speed or acceleration affects tracking error and vibration.

3. Optimize with RL: Wrap the learned model as a Gymnasium environment (a skeleton is provided). Train an RL agent to find motion parameters that make the robot move as fast as possible while minimizing vibrations. The reward function balances cycle time against vibration metrics.

Two vibration metrics are used (code provided):

a) Peak overshoot: the maximum deviation beyond the target position after arriving. Captures the worst-case oscillation spike.

peak = max(|actual(t) − target|) for t in settling window

b) RMS of position error: the root-mean-square of the position error during the settling window. Captures sustained vibration and lingering wobble.

rms = sqrt(mean((actual(t) − target)^2)) for t in settling window

Students may combine these metrics with custom weights, add their own metrics (e.g. jerk, filtering code provided as reference), or propose alternative formulations. Validate the RL-optimized policy on the real UR10e and compare before/after.

Keywords: Sim-to-real, reality gap, reinforcement learning, trajectory optimization, vibration minimization, RTDE

## Tiers

**Bronze:** Load the provided RTDE dataset. Visualize target vs actual joint trajectories for different speed and acceleration settings. Compute peak overshoot and RMS position error across all runs using the provided metric code. Present findings in clear plots that show which parameter combinations produce the most and least vibration.

**Silver:** Train a gap model that predicts actual joint positions given the target trajectory and motion parameters (speed, acceleration). A linear regression baseline is provided. Beat it using any architecture (MLP, random forest, polynomial fit, etc.). Evaluate prediction accuracy on held-out runs and report per-joint error statistics.

**Gold:** Integrate the trained gap model into the provided Gymnasium environment skeleton. Train an RL agent (using Stable-Baselines3 or similar) that selects speed and acceleration to maximize motion speed while minimizing vibration. Show that the learned policy outperforms fixed-parameter baselines on the vibration metrics.

**Diamond:** Pick one: (A) Extend the RL approach with advanced techniques: multi-objective optimization, curriculum learning, per-joint parameter selection, or additional state features, and demonstrate measurable improvement over the Gold policy. Or (B) transfer the learned policy to the real UR10e, record trajectories, and compare real-robot metrics against gap model predictions. Or do both. Or do both.

## Tools, methods and materials

**Data:** RTDE recordings from a real UR10e consisting of multiple runs with systematically varied speed, acceleration, and blend settings, capturing targets and actuals for joint positions, velocities, accelerations, currents, TCP pose, and force/torque. Provided as structured datasets.

**Analysis and modeling:** Python data science stack (NumPy, Pandas, Matplotlib for visualization, scikit-learn or PyTorch for the gap model).

**RL training:** Stable-Baselines3 or a similar RL library. A Gymnasium environment skeleton is provided for wrapping the learned gap model. Students define the

observation space, action space, reward function, and training loop themselves.

**Vibration analysis:** code for computing peak overshoot and RMS position error is provided. Jerk computation with Butterworth filtering is included as optional reference code.

**Development environment:** URSim Docker container (provided via docker-compose) for testing scripts and generating target trajectories. All development happens on the student's own laptop.

**Validation:** two UR10e robots are available for running optimized policies on real hardware and comparing before/after metrics.

From UR, the team will receive a docker-compose setup for URSim, the RTDE dataset (CSV files with documentation on all signals), a Python RTDE client library, a linear regression baseline model, vibration metric code (peak overshoot, RMS error, plus optional jerk with Butterworth filtering), a Gymnasium environment skeleton, and a data recording script. Two UR10e robots are available for real-hardware validation. UR will be available to discuss details along the way.

View pdf

Back to industry cases