COM SCI 188: Introduction to Robotics (winter 2026)

Coding Assignment 1:

PID Controllers for Robosuite Manipulation Tasks

Deadline: 1/23 11:59pm

Overview

In this assignment, you will develop a foundational control system for robotic manipulation. You are required to implement a **Proportional-Integral-Derivative (PID) controller** and design **hard-coded policies** to solve three distinct manipulation tasks within the robosuite simulation environment.

For all tasks, assume the environment provides **perfect state observability**. This means the 3D poses (position and orientation) of all relevant objects—blocks, nuts, pegs, and door handles—are directly accessible through the state observation vector.

Submission & Requirements

- **Code:** Use the provided skeleton code and do not rename classes.
- **Files:** Submit `pid.py` and `policies.py` via Gradescope.
- **Resources:** Refer to the Robosuite Documentation for API specifics and observation space details.
- **Make sure you read this entire document before starting!**

COM SCI 188: Introduction to Robotics (winter 2026)

Part 1: PID Controller Implementation (pid.py)

Your first goal is to design a **task-agnostic** PID controller. This controller will bridge the gap between high-level target poses and low-level robot joint or end-effector commands.

Understanding the PID Components

A PID controller calculates an "error" value as the difference between a desired setpoint (target pose) and a measured process variable (current pose). It then applies a correction based on three terms:

- **Proportional (P):** Produces an output proportional to the current error value. A high proportional gain (K_p) results in a large change in output for a given change in error.
- **Integral (I):** Accounts for past values of the error. If the error persists over time, the integral term increases, helping to eliminate residual steady-state errors.
- **Derivative (D):** Predicts future error based on its current rate of change. This term provides a damping effect, reducing overshoot and improving settling time.

Your implementation should be robust enough to drive the robot arm to any specified pose in 3D space, regardless of the specific manipulation task being performed.

Part 2: Task-Specific Policies (policies.py)

Once your PID controller is functional, you will write control policies to solve three specific challenges. Each policy must translate a high-level goal into a sequence of **generalizable waypoints**.

Strategic Waypoint Planning

Success in these tasks depends on how you define your waypoints. Since object locations are **randomized at the start of each episode**, you cannot hard-code absolute coordinates. Instead, you must calculate waypoints **relative to the observed object poses**.

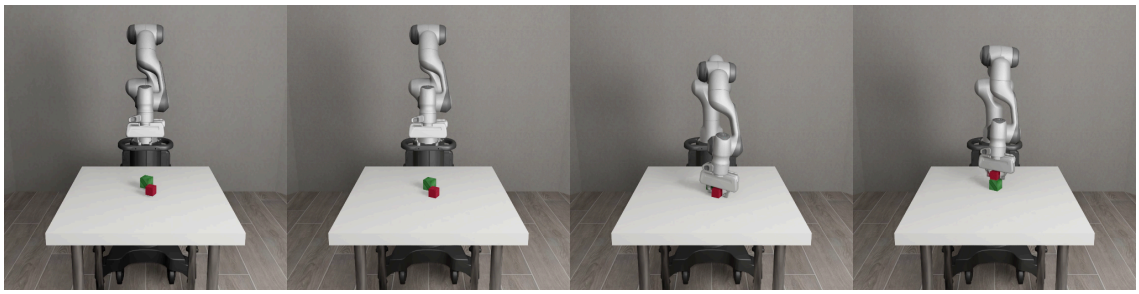
For example, to pick up a block, your policy might sequence through:

1. **Hover:** A point directly above the object's current position to avoid collisions.
2. **Approach:** Lowering the gripper to the object's exact pose.
3. **Grasp:** Actuating the gripper.
4. **Lift:** Moving to a safe height before proceeding to the goal location.

COM SCI 188: Introduction to Robotics (winter 2026)

The Three Tasks (2 Points Each)

Block Stacking:



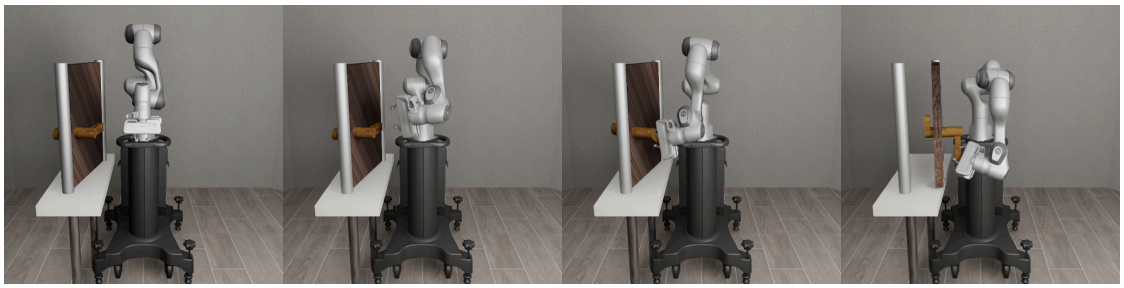
- **Scene Description**: Two cubes are placed on the tabletop in front of a single robot arm.
- **Goal**: The robot must place one cube on top of the other cube.
- **Start State Distribution**: The cube locations are randomized at the beginning of each episode.

Nut Assembly (2 points):

- **Scene Description**: Two colored pegs (one square and one round) are mounted on the tabletop, and two colored nuts (one square and one round) are placed on the table in front of a single robot arm.
- **Goal**: The robot must fit the square nut onto the square peg and the round nut onto the round peg. This task also has easier single nut-and-peg variants.
- **Start State Distribution**: The nut locations are randomized at the beginning of each episode.

COM SCI 188: Introduction to Robotics (winter 2026)

Door Opening (2 points):



- **Scene Description**: A door with a handle is mounted in free space in front of a single robot arm.
- **Goal**: The robot arm must learn to turn the handle and open the door.
- **Start State Distribution**: The door location is randomized at the beginning of each episode.

COM SCI 188: Introduction to Robotics (winter 2026)

Policy on AI Usage for Function Generation

In this assignment, you are permitted to use **Generative AI** (such as Gemini or ChatGPT) to assist in writing specific, low-level functions. While the core logic of the PID controller and high-level strategy must be your own work, AI can be used to streamline boilerplate code or mathematical transformations.

To maintain academic integrity and clarity, any AI-generated function must be accompanied by a **standardized function header** and a **declaration statement**.

Standard Function Header Format

For every function generated or significantly assisted by AI, use the following Python docstring format. This helps both you and the graders understand the function's inputs, expected behavior, and limitations.

```
def function_name(parameter_1, parameter_2):
    """
    [Short Description]: A concise summary of what the function does.
    [AI Declaration]: Generated using [AI Tool Name] with the prompt: "[Your Prompt]"

    Args:
        parameter_1 (type): Description of the first input.
        parameter_2 (type): Description of the second input.

    Returns:
        type: Description of the return value.

    Notes:
        Any manual modifications made to the AI output or specific constraints.
    """
    # Function body here
    pass
```

COM SCI 188: Introduction to Robotics (winter 2026)

Best Practices for AI Integration

- **Verification:** You are responsible for the accuracy of AI-generated code. Always test generated functions to ensure they behave as expected.
- **Iterative Prompting:** If the AI output is incorrect, refine your prompt with more specific constraints, such as the required library (e.g., `numpy`) or the expected dimensionality of the vectors.
- **Process Documentation:** Keep a log of the prompts you used. If your policy relies heavily on AI-generated segments, you may be asked to explain the underlying logic during a code review.