README.md

# Homework2

Homework2 is the second part of midterm project. You will learn to apply the knowledge about robot dynamics.

## Getting started

Please follow the document of SAPIEN to install the environment. Currently, only Ubuntu is fully supported. MacOS is experimentally supported and feel free to report any bug. **Sapien has been updated recently. Please upgrade your version.**

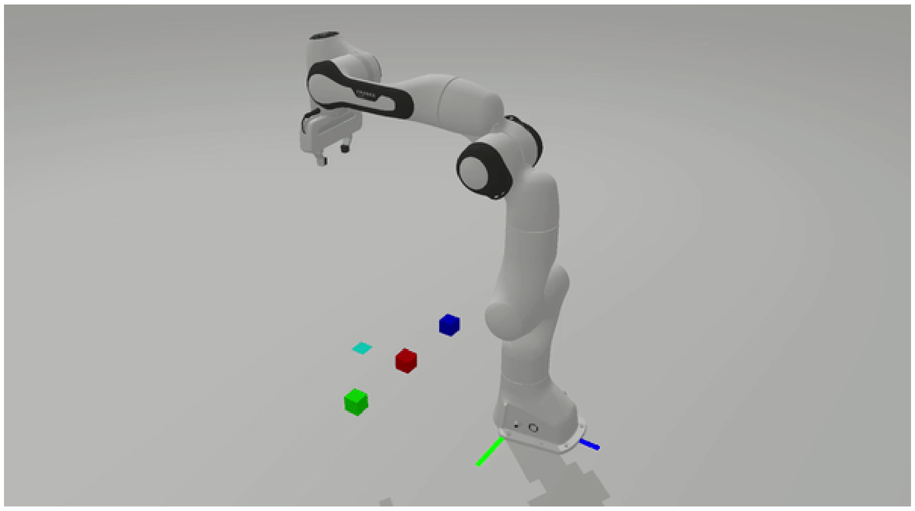
```
# Use the correct pip if you are using virtualenv or conda.
pip install sapien --upgrade  # Update to the latest sapien
```

**It is suggested to run the tutorial robotics in SAPIEN before the next step.** It is also suggested to use an editor, like VSCode or Pycharm, so that you can jump to definitions to get an idea of APIs.

## Instruction

The goal of this assignment is to achieve:

- move the end-effector dynamically to a given target pose



The starter code provides these files:

- `hw2.py` : run this file to debug and evaluate; not necessary to modify
- `hw2_env.py` : implement this file
- `stacking_env.py` : provide basic functions; not necessary to modify

### Overview

In this assignment, instead of implementing complex motion planning, we introduce a relative simple method to move the end-effector dynamically to a given target pose. The method should be implemented in `move_to_target_pose_with_internal_controller` following several steps:

1. For each time step, compute the relative transformation from the current end-effector pose to the target pose `target_ee_pose` . Note that the transformation is described in the body frame.
2. Compute the exponential coordinate of the relative transformation by `pose2exp_coordinate` . The exponential coordinate can be decomposed into the `unit_twist` and the angle `theta` . Given the time left to approach the target, you can compute the average `body_twist = unit_twist * (theta / time_to_target)` .
3. Convert the body twist to the spatial twist. Note that the spatial frame is the same as the robot base frame.
4. Compute the joint velocities `qvel` from the spatial twist by `compute_joint_velocity_from_twist` . Jacobian is provided in the function.
5. Call `internal_controller` to execute the joint velocities `qvel` .
   > Concretely, it computes the target joint poses `qpos` by adding `qvel * timestep` to the current joint poses. Given the target joint poses and velocities, the Sapien engine can compute the generalized force `qf` to achieve the velocities.

### Pick and place boxes

`pick_up_object_with_internal_controller` should be implemented to pick up a box. Similar to homework1, you should compute the position of the box given its point cloud. Then you need to call `move_to_target_pose_with_internal_controller` to move the end-effector to approach the box, and call `close_gripper` to grasp the box. Next you need to lift the box to a certain height (move the end-effector along the z-axis).

`place_object_with_internal_controller` should be implemented to place the box to the target position. Concretely, you need to move the end-effector so that the box is located at the target position, and call `open_gripper` to release the box.

### Implement your own controller

You can use the internal controller (provided for you) to stack the first two box (red and green). But for the third box (blue), instead of using the internal controller, you need to implement your own controller and use it in `move_to_target_pose_with_user_controller` . Accordingly, you can implement `pick_up_object_with_user_controller` and `place_object_with_user_controller` with your controller. It is relatively hard to directly control the dynamics of the robot. You will need some time on the parameters of your controller (e.g. PID).

## Grading

The assignment will be evaluated by running hw2.py to check the correctness. The detailed rubric is listed as follows:

- Implement `move_to_target_pose_with_internal_controller` correctly: 50%
- Successfully pick and place two boxes with the internal controller: 30%
- Successfully pick and place the last box with the user controller: 20%

It is not necessary to import extra libraries. You will also lose points if you use `scipy` and `transform3d` . Late submission will also lose points. For the second homework, even opencv-python and ikfast-pybind is not necessary.

## Turning it in

**The deadline of the homework2 is May 17th, 12 p.m.** For homework2, you only need to submit hw2_env.py to gradescope. You are not required to submit a report. One submission for each team.

> Gradescope entry code: M5WDJG

## Academic integrity

You are allowed to work in teams, but are not allowed to use the answers developed by others. You are not allowed to copy online resources, like GitHub repositories. Please ask the instructor first if you are not sure whether you can refer to some resources.

If the work you submit is determined to be other than your own, you will be reported to the Academic Integrity Office for violating UCSD's Policy on Integrity of Scholarship.