Advanced Robotic Manipulation (CS 327A) Homework #6

(Spring 2022/2023) Due: **Friday, May 26, 11:59pm**

In this homework you will explore accomplishing multiple tasks through hierarchical control schemes and obstacle avoidance via artificial potential fields.

Example Videos

You can find the example video for the SAI portion of this homework at this link: HW 6 Example Video

You must be logged into your Stanford Google account to access the videos.

Code Submission Instructions

For Problem 1e, submit your controller files to the Gradescope assignment.

Additionally, you should record a video of the simulation running in each part, upload this video somewhere accessible (Google Drive or Youtube both work well), and then include a link to the video for each part.

Alternatively, you may come by Wesley's office hours or email him to schedule a Zoom session in order to show your working simulations.

Code Update

You can find the new homework code by navigating to the cs327a directory in your computer and following the steps below:

1. Stash your changes and pull the new updates (which will include the solutions) by using the following commands:

```
$ git stash
$ git pull
```

2. Retrieve your own changes (that you previously stashed) by typing:

```
$ git stash pop
```

3. Build the updated repo by navigating to cs327a/build and running:

```
$ cmake .. && make
```

4. The new code is available in the hw6 folder

   NOTE: Like last week's HW5, this week's homework uses a legacy application development paradigm for SAI2 simulations, so you will only see one `.cpp` source file. It contains both the simulation and the control threads. Functionally, this won't make a difference for your implementation of the controller. Just place your code in the fill-in section as indicated.

Problem 1 - Artifical Potential Fields and Whole Body Control

For this problem, you will implement obstacle avoidance on the familiar PUMA robot as part of a hierarchical whole body control scheme. The robot's goals are decomposed into three parts:

1. Avoiding a moving obstacle by keeping a safe distance
2. Tracking an end-effector position trajectory
3. Maintaining a desired joint posture

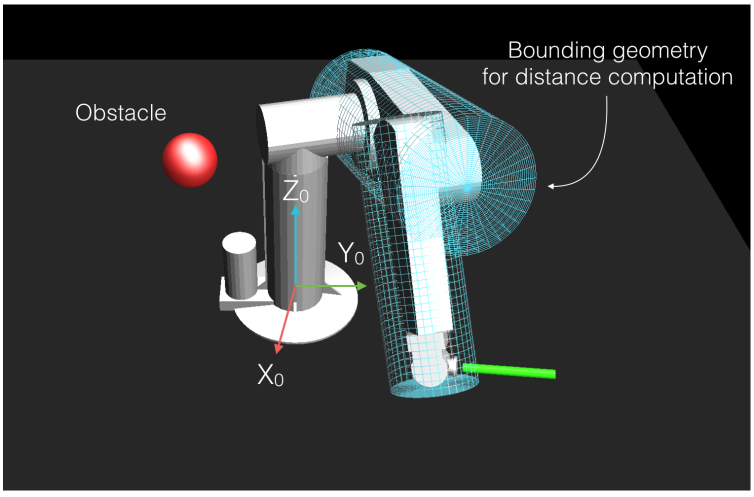
Let the total applied control torque be Γ. We will design three separate controllers Γ_c, $\Gamma_{t|c}$ and $\Gamma_{p|t|c}$, where:

- Γ_c is the control torque applied to maintain a safe distance and avoid the obstacle which we treat as a constraint
- $\Gamma_{t|c}$ is the control torque applied to track the desired end-effector position trajectory while not violating the constraint
- $\Gamma_{p|t|c}$ is the control torque applied to maintain a desired joint posture while not violating the constraint and not disturbing the task (constraint and task consistent posture controller)

Putting together the individual controllers, we will obtain the following torques (where g is the overall feed-forward compensation for gravity):

$$\Gamma = \Gamma_c + \Gamma_{t|c} + \Gamma_{p|t|c} + g$$

Shown below is a snapshot of the setup:



(a) Setup

After you complie the starter code, run the `hw6_p1` file and try moving the sphere around. The robot should simply fall and wiggle around as no control torques are currently being applied.

**** Keyboard control of obstacle ****

The red sphere visible when you run the starter code represents the moving obstacle that the Puma

must avoid. It is implemented as a phantom object with no physical properties that can be moved around in the Y_0 - Z_0 plane with arrow keys. `Left` and `Right` arrow keys move the obstacle along the Y_0 axis, whereas the `Up` and `Down` arrow keys move the obstacle along the Z_0 axis.

(b) Constraint controller design

The constraint considered in this problem is to maintain a safe distance from the red sphere. One way to do so is to create an artificial potential field.

Consider:

- $d \rightarrow$ the distance from the surface of the sphere to the closest link i on the PUMA
- $x_{ci} \rightarrow$ the location of the point on link i closest to the sphere
- $c \rightarrow$ the location of the center of the sphere

**** Write the expression for the potential function $U_c(d)$ ****

The gradient of this potential field should produce a repulsive force field in the direction $n_c = (c - x_{ci})/\|c - x_{ci}\|$.

One possible design is presented on Slide 25 of the 05-12-21 Elastic Planning lecture handout. Choose a cut-off distance d_0 such that the potential as well as its gradient are zero for $d \geq d_0$.

When implementing the constraint control torque, we consider the constraint "active" if $d < d_0$ when the sphere gets too close to the PUMA, else we consider it "inactive". When the constraint is "inactive", we set $\Gamma_c = 0$. When the control is "active", the control torques to maintain a safe distance from the sphere can then be written as:

$$\Gamma_c = J_c^T \nabla U_c + J_c^T \Lambda_c (-k_{vc} J_c \dot{q})$$

where the additional term on the right dampens motions in the constraint direction. J_c is the constraint Jacobian given by

$$J_c = n_c^T J_v (x_{ci})$$

where $J_v (x_{ci})$ is the linear velocity Jacobian at the point x_{ci} on link i.

(c) Task controller design

For this section, we consider the task of maintaining the end-effector of the PUMA along the following desired trajectory:

$$\begin{aligned} x_{ee,d} &= 0.7 \\ y_{ee,d} &= 0.2 + 0.4 \sin\left(\frac{2\pi t}{6}\right) \\ z_{ee,d} &= 0.5 \end{aligned}$$

**** Write the expression for $\Gamma_{t|c}$ **** by designing a PD controller such that the end-effector follows the above trajectory while not violating the constraint. That is, if the constraint is "active", the task control torque should not produce any acceleration in the constraint direction. Mathematically: $J_c A^{-1} \Gamma_{t|c} = 0$.

Notes

- The constraint consistent task space inertia $\Lambda_{t|c}$ is given by $\left(J_{t|c}A^{-1}J_{t|c}^T\right)^{-1}$
- When implementing this controller in code, note that:
  - If the constraint is "active", then $J_{t|c} = J_t N_c$
  - Else if the constraint is "inactive", then $J_{t|c} = J_t I$ where I is the identity matrix
- The end-effector tip is located at $[-0.2 \;\; 0 \;\; 0]$ in the last joint frame, and is available through the variable `ee_pos_local` in the starter code.

Extra Credit: The above controller will work as expected, but it can be made better. Currently, we allow the constraint torques Γ_c to disturb the task by producing accelerations given by $J_t A^{-1}\Gamma_c$ which are non-zero as long as the constraint is "active". Try to modify F^*_{motion} to compensate for these accelerations.

(d) Posture controller design

Finally, design a posture PD controller $\Gamma_{p|t|c}$ that holds the following joint angles:

$$q_d = \begin{bmatrix} 0.00 \\ 5.90 \\ 3.70 \\ 1.57 \\ 1.75 \\ 3.14 \end{bmatrix}.$$

Ensure that the posture control torques produce no acceleration either in the constraint or in the task co-ordinates. That is, we require:

$$J_c A^{-1} \Gamma_{p|t|c} = 0 \quad \text{and} \quad J_t A^{-1} \Gamma_{p|t|c} = 0$$

(e) Implementation

Now implement your controller in the portion marked as `FILL ME IN` in `cs327a/hw6/hw6_p1.cpp`. Once you have your controller running, try moving the sphere around to test its robustness and make a video or show at office hours.

**** Note ****

The distance computations are implemented not directly between the mesh representations of the PUMA links and the sphere, but instead with bounding cylinders around the links. This is to improve computational efficiency. We also consider only the two largest links of the PUMA (the upper arm and the lower arm).

**** Hint ****

You should observe that the robot changes its posture to avoid the obstacle while keeping the end-effector on the desired trajectory. However if the obstacle is in such a position that the task is unachievable, the controller prioritizes avoiding the obstacle over performing the task, as designed. Also, if the obstacle is removed from the vicinity of the robot, the robot goes back to a desirable posture with the elbow pointing up.