Assignment 2

Assignment Objective

In this assignment, our objective is to learn how to evaluate grasp restraints. Step by step, we will write a program that takes in contact locations on an object and evaluates force closure.

Instructions

In this assignment, we will ask you to fill out missing pieces of code in a Python script. You will submit the completed code to autograder. Your code is passed to an auto-grader that will verify the correctness of your work and assign you a score out of 100.

- Download the assignment script `assignment_2.py` from the Files menu of Canvas.
- For each "Part" of the HW assignment, fill in the missing code as described in the problem statement.
- Submit the completed code to Canvas.
- The contribution of each part to your total score is noted in the subtitle.
- The assignment is due on October 16^{th}, 2023.
- You will need `numpy` and `cvxpy` libraries.

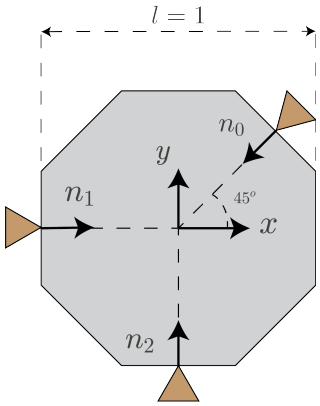

Figure 1. Does this grasp have form closure? Assume that the coefficient of friction $\mu = 0.3$.

Part 1 – Contact Locations and Normals (20 points)

Consider the grasp of a regular octagon depicted in Fig. 1. There are a total of 3 contacts, each with a corresponding contact location $\boldsymbol{r}_i$ and contact normal $\boldsymbol{n}_i$ for $i = 0, \cdots, 2$. In this part, we will recover the contact locations $\boldsymbol{r}_i$ and contact normals $\boldsymbol{n}_i$ in the object frame. Let's denote:

$$\boldsymbol{r}_i = \begin{bmatrix} r_x \\ r_y \end{bmatrix}, \quad \boldsymbol{n}_i = \begin{bmatrix} n_x \\ n_y \end{bmatrix}$$

where the subscripts x and y denote the projections of the vector along the object frame axes. Let's define the contact location matrix and contact normal matrix as:

$$\mathbf{R} = \begin{bmatrix} \boldsymbol{r}_0 & \boldsymbol{r}_1 & \boldsymbol{r}_2 \end{bmatrix}_{2\times 3}$$
$$\mathbf{N} = \begin{bmatrix} \boldsymbol{n}_0 & \boldsymbol{n}_1 & \boldsymbol{n}_2 \end{bmatrix}_{2\times 3}$$

Complete the function `get_contacts` in `assignment_2.py` to return these two matrices for the grasp depicted in Fig. 1; i.e. this function returns two matrices but does not require an input:

$$\mathbf{R}, \mathbf{N} = \text{get\_contacts}()$$

Part 2 – Contact Jacobians and the Grasp Matrix (20 points)

Now that we have the contact locations and the contact normals, we would like to compute the contact Jacobians. Recall that the contact Jacobian for the planar case is written as:

$$\mathbf{J}_i = \begin{bmatrix} n_y & n_x \\ -n_x & n_y \\ -r_x n_x - r_y n_y & r_x n_y - r_y n_x \end{bmatrix}_{3\times 2}$$

where the first column is the tangential component and the second column is the normal component. The grasp matrix is built from the contact Jacobians as:

$$\mathbf{G} = \begin{bmatrix} \mathbf{J}_0 & \mathbf{J}_1 & \mathbf{J}_2 \end{bmatrix}_{3\times 6}$$

Complete the function `calculate_grasp` in `assignment_2.py` such that the inputs are $\mathbf{R}$ and $\mathbf{N}$ (which you calculated in the previous part) and the output is the Grasp matrix:

$$\mathbf{G} = \text{calculate\_grasp}(\mathbf{R}, \mathbf{N})$$

Part 3 – Friction Cone Facet Normals (20 points)

We can define each friction cone using the coefficient of friction $\mu = \tan\theta = 0.3$ and the facet normal vector as for each contact i as:

$$\mathbf{F}_i = \frac{1}{\sqrt{1+\mu^2}} \begin{bmatrix} 1 & \mu \\ -1 & \mu \end{bmatrix}$$

We can define the Friction Facet Normal matrix for the grasp depicted in Fig. 1 as:

$$\mathbf{F} = \text{BlockDiag}\{\mathbf{F}_0, \mathbf{F}_1, \mathbf{F}_2\}$$

Complete the function `calculate_facet` in `assignment_2.py` such that the output is $\mathbf{F}$:

$$\mathbf{F} = \text{calculate\_facet}(\mu)$$

We will use this matrix as part of constraints to analyze force closure in subsequent parts.

Part 4 – Grasp Rank (5 points)

Now we'd like to know if the grasp qualifies as a form or force closure. The first step is to check if the grasp rank is 3. Complete the function `compute_grasp_rank` in `assignment_2.py` to evaluate the grasp rank and return `True` if the rank is equal to 3, else return `False`:

$$\text{flag} = \text{compute\_grasp\_rank}(\mathbf{G})$$

where flag is either `True` or `False`. Hint: The `numpy` linear algebra library has a function for matrix ranks, feel free to use it.

Part 5 – Grasp Constraints and Force Closure Test (35 points)

Once we have evaluated whether the grasp is rank sufficient, we can test for force closure using the following linear program:

$$\begin{aligned} \textbf{LP:} \quad & \text{maximize} && d \\ & \text{s.t.} && \mathbf{G}\boldsymbol{f}_c = 0 \\ & && \mathbf{F}\boldsymbol{f}_c - \mathbf{1}d \geq 0 \\ & && d \geq 0 \\ & && \boldsymbol{e}^T\boldsymbol{f}_c \leq n_c \end{aligned}$$

where $\mathbf{1}^T = [1, \cdots, 1]_{6\times 1}$, $\boldsymbol{e}^T = [0, 1, 0, 1, 0, 1]$, and n_c is the number of contacts. We can re-write the linear program as:

$$\begin{aligned} \textbf{LP:} \quad & \text{maximize} && \boldsymbol{c}^T\boldsymbol{x} \\ & \text{s.t.} && \mathbf{A}\boldsymbol{x} = \boldsymbol{b} \\ & && \mathbf{P}\boldsymbol{x} \leq \boldsymbol{q} \end{aligned}$$

where:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{f}_c \\ d \end{bmatrix}_{7\times 1}$$
$$\boldsymbol{c}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{1\times 7}$$

Complete the function `compute_constraints.py` in `assignment_2.py` that returns $\mathbf{A}$, $\boldsymbol{b}$, $\mathbf{P}$, $\boldsymbol{q}$, $\boldsymbol{c}$:

$$\mathbf{A},\ \boldsymbol{b},\ \mathbf{P},\ \boldsymbol{q},\ \boldsymbol{c} = \text{compute\_constraints}(\mathbf{G}, \mathbf{F})$$

The function `check_force_closure` in `assignment_2.py` evaluates the linear program in Part 5 and returns d^*. You do NOT need to code the linear program, only pass on the constraints you just computed. Test the force closure by passing on the constraints to `check_force_closure`:

$$d^* = \text{check\_force\_closure}(\mathbf{A},\ \boldsymbol{b},\ \mathbf{P},\ \boldsymbol{q},\ \boldsymbol{c})$$

If everything has gone well, you will notice that the grasp does satisfy force closure.

NOTE: In this case, $\boldsymbol{f}_c = \begin{bmatrix} f_{t,0} & f_{n,0} & f_{t,1} & f_{n,1} & f_{t,2} & f_{n,2} \end{bmatrix}^T$