EECS C106B Project 4: Grasp Planning with Sawyer *

Due date: Monday, April 12th at 11:59pm

Goal

Grasp Analysis and Planning

The purpose of this project is to combine many of the topics presented in this course to analyze grasps planned for and executed by a Sawyer with a parallel-jaw gripper. You will also implement your own grasp-planning algorithm and test it out on two custom objects.

Contents

*Developed by Jeff Mahler, Spring 2017. Expanded and edited by Chris Correa and Valmik Prabhu, Spring 2018, Spring 2019. Further expanded and developed by Amay Saxena and Tiffany Cappellari, Spring 2020.

1 Theory

Here's a quick refresher on grasp theory drawn from [1]. We can define a contact as

$$F_{c_i} = B_{c_i} f_{c_i}$$

Where B is the contact basis, or the directions in which the contact can apply force, and f is a vector in that basis. F is the wrench which the contact applies. In our case, we use a soft contact model, which has both lateral and torsional friction components, so the basis is

$$B_{c_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

However, in the real world, friction is not infinite. For the contact to resist a wrench without slipping, the contact vector must lie within the *friction cone*, which is defined

$$FC_{c_i} = \{f \in \mathbb{R}^4 : \sqrt{f_1^2 + f_2^2} \leq \mu f_3, f_3 > 0, |f_4| \leq \gamma f_3\}$$

However, we want the wrenches that a contact point can resist in the world frame, not the contact frame. So we define

$$G_i := Ad^T_{g_{oc_i}^{-1}} B_{c_i}$$

A grasp is a set of contacts, so we define the wrenches (in the world frame) a grasp can resist as:

$$F_o = G_1 f_{c_1} + \cdots + G_k f_{c_k} = \begin{bmatrix} G_1 & \cdots & G_k \end{bmatrix} \begin{bmatrix} f_{c_1} \\ \vdots \\ f_{c_k} \end{bmatrix} = Gf$$

A grasp is in *force closure* when finger forces lying in the friction cones span the space of object wrenches

$$G(FC) = \mathbb{R}^p$$

Essentially, this means that any external wrench applied to the object can be countered by the sum of contact forces (provided the contact forces are high enough).

1.1 Discretizing the Friction Cone

All of the grasp metrics we use are structured as optimization problems with $f \in FC$ as a constraint. However, set of constraints that make up the friction cone for contact i

$$FC_{c_i} = \begin{cases} \sqrt{f_1^2 + f_2^2} \leq \mu f_3 \\ f_3 > 0 \\ |f_4| \leq \gamma f_3 \end{cases}$$

contains a quadratic constraint. Thus, none of the optimization problems will be convex. In order to guarantee a solution, you can approximate the (conical) friction cone as a pyramid with n vertices. The level sets of the friction cone are circles, but the level sets for this convex approximation are n sided polygons circumscribed by the circle. Thus, the interior of this convexified friction cone is a conservative approximation of the friction cone itself.
Any point in the interior of this pyramid can be described as a sum of

$$f = \alpha_0 f_0 + \sum_{i=1}^{n} \alpha_i f_i = F\alpha$$

where f_i are the edges of the pyramid and f_0 a straight line in z, and the weights α are all non-negative. Here, we can write a composite matrix F with the f_i vectors as its columns.

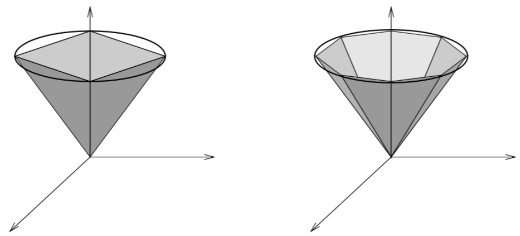


Figure 1: Approximations of the friction cone. From section 5.3 of [1]

Thus you can represent any point in the friction cone as a set of linear constraints with these new grasp forces. One thing you should note is that the magnitude of $f^{\perp}$ is no longer just α_0. You'll have to pay close attention to the way that you define your f_i so that you'll be able to calculate $f^{\perp}$ from α. You'll also have to figure out how to add the soft contact constraint ($|f_4| \leq \gamma f_3$) into your new approximate friction cone.

With this approximation, the condition that $f \in FC$ is equivalent to the pair of linear constraints $\{f = F\alpha, \alpha \geq 0\}$ (where this inequality is understood to be element-wise). In most applications, you can turn a condition in terms of f with the constraint $f \in FC$ into a condition in terms of α with the constraint $\alpha \geq 0$, by substituting f with $F\alpha$. This is a much more tractable constraint and will turn all of our optimization problems into convex problems that can be easily solved by any convex optimization solver.

1.2 Grasp Metrics

When designing a grasp, you'll need some way of determining its quality. Force closure is helpful, but it's a binary metric (yes it's in force closure or no it isn't) so it's often unhelpful in ranking grasps. Instead, you'll be implementing three common grasp metrics: Gravity Resistance, Ferrari-Canny, and Robust Force Closure.

1.2.1 Gravity Resistance

When grasping objects, often the largest force you'll have to counteract is gravity, so the gravity resistance metric calculates how well your grasp will be able to resist the force of gravity. To define this metric, simply compute

$$\begin{aligned} \widehat{f} = \underset{f}{\operatorname{argmin}} & \, \|f\|_2^2 \\ st. \; & f \in FC \\ & Gf = -W_g \end{aligned}$$

where W_g is the wrench on the object's center of mass due to gravity. The metric is

$$J = \sum_i \widehat{f}_{3_{c_i}} = \sum_i \widehat{f}^{\perp}_{c_i}$$

Where $\widehat{f}_{3_{c_i}}$ is the perpendicular finger force at contact i, and is also denoted as $\widehat{f}^{\perp}_{c_i}$. Remember that $\widehat{f}_{3_{c_i}}$ is constrained to be positive (by the friction cone), so this cost will always be positive. This metric calculates the minimum finger forces needed to resist gravity, and a *lower* cost indicates a *more secure* grasp. If the grasp cannot resist the force of gravity (the problem is infeasible), then the cost is infinity.

Once we incorporate the discretized approximation of the friction cone, this optimization problem becomes a quadratic program. There are some ways of simplifying it further by incorporating the discretization of the friction cone directly into the problem and optimizing over $\alpha \geq 0$. We leave the details of these simplifications up to you. You may use your favourite convex optimization solver (we recommend `cvxpy` or `casadi`).

1.2.2 Ferrari-Canny

The Ferrari-Canny metric [2] is an extension to the force closure metric that captures how much grasp effort is needed to maintain force closure. The paper defines the metric as follows. First, define a *local quality metric* LQ, a function of the wrench space, as

$$\begin{aligned} LQ(w) = \max_f \ & \frac{\|w\|}{\|f\|} \\ st.\ & f \in FC \\ & Gf = w \end{aligned}$$

$LQ(w)$ is a measure of how efficiently a given wrench w can be resisted. A wrench is resisted more efficiently if the norm of the input force (at the contact points) required is low. Then, we measure the total quality of the grasp as the worst case efficiency ratio.

$$Q = \min_w LQ(w)$$

We consider the worst case efficiency since we don't, in general, have control over the wrench acting on the body that must be resisted.

This metric is difficult to implement as written, so we will perform some simplifications. First, notice that the the magnitude of the applied wrench scales linearly with the magnitude of the input force vector. Since the grasp metric is the ratio between these two magnitudes, we can equivalently optimize over unit wrenches. Then, LQ can be re-written as

$$\begin{aligned} LQ(\widehat{w}) = \max_f \ & \frac{1}{\|f\|} \\ st.\ & f \in FC \\ & Gf = \widehat{w} \end{aligned}$$

defined over unit 6D vectors $\widehat{w}$. Maximizing $1/\|f\|$ is equivalent to minimizing $\|f\|$. So we define a new local quality metric

$$\begin{aligned} \widehat{LQ}(\widehat{w}) = \min_f \ & \|f\|_2^2 \\ st.\ & f \in FC \\ & Gf = \widehat{w} \end{aligned}$$

where we have made two important changes. First, we have reciprocated the objective function. Secondly, we have turned the norm into a norm-squared. These two changes do not change the optimal force f, but they do change the value of the objective function. So our final quality metric is

$$Q = \min_{w, \|w\|=1} \frac{1}{\sqrt{\widehat{LQ}(w)}}$$

The advantage of making this change is that now the optimization problem $\widehat{LQ}$ (after incorporating the standard discretization of the friction cone) is a quadratic program and can be solved by any convex optimization solver.

It remains to discuss how to solve the outermost problem Q. Indeed, this overall problem is difficult to solve exactly. So we can find an approximate solution through Monte-Carlo sampling. To implement this metric, we recommend first sampling N 6D unit vectors $\{\widehat{w}_i\}_{i=1}^N$ randomly from the unit sphere, finding $\widehat{LQ}(\widehat{w}_i)$ for each $\widehat{w}_i$, and then picking the smallest $1/\sqrt{\widehat{LQ}}$. This is a randomized approach, so you will get a random result, but by increasing the number of samples N you can reduce the variance of your estimate. You can sample from the unit sphere by first sampling from a rectangle and then normalizing the result. We recommend starting with $N = 1000$, and increasing it if you see high variability. Note that your grasp metrics do not have any tight runtime requirements.

Note that once again, you will find it useful to bring in the discretization of the friction cone into the problem directly and optimizing over α.

You can also think about the Ferrari-Canny metric geometrically. If you define the friction cone when $\|f^\perp\|$ is 1, the Ferrari-Canny metric is the minimum distance between the origin and the convex hull of the friction cone. It's possible to efficiently compute this metric geometrically by examining the polytope you get when you take the level set of the friction cone. However, the implementation is quite involved without using specialized geometry packages, hence the solution above.

1.2.3 Robust Force Closure

This is the grasp metric used in the DexNet paper that you can find here.

The idea behind the Robust Force Closure grasp metric is to quantify *to what extent* a grasp is force closure. For instance, you may imagine that we have two force closure grasps. Naively, there is no way to tell these apart, even though it may be the case if one of the contacts moved even a little bit while executing grasp 1, it would fail to be force closure, whereas with grasp 2, the contacts can handle a fair bit of disturbance before the grasp ceases to be force closure. We would like to quantitatively say that grasp 2 is "better" than grasp 1.

So, instead of simply checking whether a given grasp is force closure or not, we instead introduce some random noise to the grasp, and then ask what the *probability* is that the resultant perturbed grasp is force closure. Let's define this rigorously. We describe a grasp using a pose $\mu \in SE(3)$, which describes where the end effector of the robot should be when the grasp is executed. We additionally have $\mathcal{M} \in \mathcal{O}$, which is a model of the object we are grasping, with $\mathcal{O}$ being the space of all objects. Together, the pose μ and the model $\mathcal{M}$ can be used to check if the specified grasp is in force closure. Let $F : SE(3) \times \mathcal{M} \to \{0, 1\}$ be the proposition:

$$F(\mu, \mathcal{M}) = 1 \text{ if grasp } \mu \text{ on object } \mathcal{M} \text{ is in force closure, else } 0 \tag{1}$$

Let δ be some random disturbance. We are deliberately keeping this vague at this time since there are many ways to perturb a pose in $SE(3)$. Let $\hat{\mu} = \mu + \delta$ be a random variable that models an uncertain pose (our original pose μ with random disturbance δ). The quality Q of the grasp $(\mu, \mathcal{M})$ will be

$$Q(\mu, \mathcal{M}) = \mathbb{P}(F(\hat{\mu}, \mathcal{M}) = 1) \tag{2}$$

In practice, we would implement the above probability by Monte Carlo estimation - taking a large number of samples from the disturbance δ, perturbing μ by each sample, and then checking what fraction of the perturbed grasps are in force closure.

It remains to be discussed how we should construct the random variable δ. We are looking for some way to perturb a given grasp. One straightforward way to do this is to instead perturb the two contact points directly. Let $x, y \in \mathbb{R}^3$ be the two contact points of your grasp. Let $\delta_x, \delta_y \overset{i.i.d}{\sim} \mathcal{N}(0, I\sigma^2)$ be two independent Gaussian random variables in $\mathbb{R}^3$, and construct a new grasp $\hat{\mu}$ by picking contact points $(\hat{x}, \hat{y}) = (\pi(x + \delta_x), \pi(y + \delta_y))$ where π is an operator that projects a point in $\mathbb{R}^3$ to its nearest point on the outer surface of the object $\mathcal{M}$. You could get cleverer by sampling Gaussians from only the planes tangent to the object at the points x and y to come up with your perturbation. The starter code provides some basic utilities that can get you started in finding intersections and projections onto the meshes of objects.

Alternatively, we could find a way to perturb the pose μ as we stated in the algorithm definition. One way of doing this is to sample δ randomly from $se(3)$, and then compute

$$\hat{\mu} = \mu \cdot e^{\hat{\delta}} \tag{3}$$

For instance, you may sample δ as a zero mean Gaussian with uniform variance σ^2 in $\mathbb{R}^6$. This is how robust force closure is implemented in the DexNet paper. Beware, however, that if you do it in this way, then the random variable $\hat{\mu}$ in fact does *not* have expectation μ. Nevertheless, this is a commonly used noise model for $SE(3)$ valued variables.

The one parameter left to tune is the variance σ^2 of the disturbance. We want to pick a variance that reasonably mimics the variation in pose in the real world when a grasp is executed multiple times. If the variance is too low, then practically every sampled grasp will be force closure if the original grasp was. If the variance is too high, then the metric will not be representative of disturbances in the real world.

When implementing this grasp metric, feel free to use either of the above techniques, or design your own disturbance. You should discuss in your report how you went about implementing your metric.

2 Project Tasks

For this project, we have planned five grasps for each object pictured in Figure 2 and have executed them on a Sawyer and recorded the data for you. We performed five trials for each of the five grasps for each of the two objects,

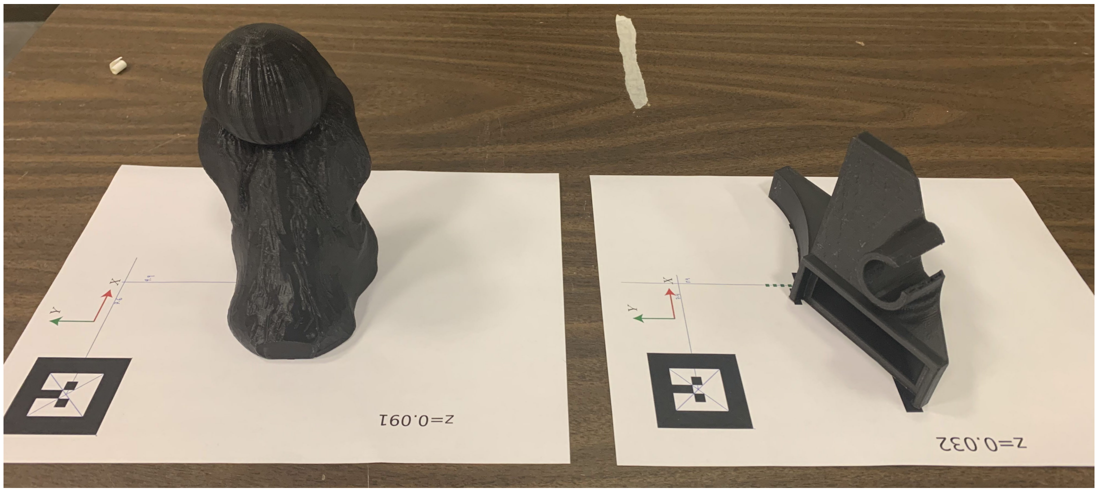


Figure 2: The two test objects with placement templates (pawn and nozzle).

and saved data about whether each of the 50 trials were a successful or failed grasp. See the starter code section for details on the data.
For visualization purposes, we also recorded video of one trial for each of the five grasps for each of the two objects. Videos of the pawn grasps can be found here. Videos of the nozzle grasps can be found here. Your job will be to write metrics to score each of the grasps. Additionally, you will design your own grasp planning algorithm, and explain its working and effectiveness by visualizing the grasps it generates against meshes of the two objects.

Write code for each of the following metrics. Use a soft finger contact models with $\mu = 0.5$ and $\gamma = 0.1$:

1. Resistance of gravity assuming an object mass of 0.25 kg and center of mass at the origin
2. Ferrari Canny
3. Robust Force Closure

The grasps provided to you were planned using very naive algorithms, and hence will fail for many interesting reasons. By visualizing each grasp using the code provided to you, and comparing the visualizations to what you see in the video, you should discuss why you believe each of the grasps succeeded or failed in the way that it did.

Next, you will design a grasp planning algorithm of your own. Write a function that will take in a mesh of an object at hand along with two contact points on the mesh. It will then return a 4x4 rigid pose for where the gripper tip should be in order to execute a grasp with those contact points. You will also take as input attributes of the gripper, such as the maximum and minimum finger spans, and the length of the gripper itself. The function should either return a rigid transform if a successful grasp with those contacts is possible, otherwise it should return None. You may wish to consider the following when designing your algorithm:

1. If the distance between the two input contact points is larger than the maximum amount your gripper can open, then no grasp is possible.
2. If the distance between the two input contact points is smaller than the minimum threshold that your gripper can close to, then no grasp is possible.
3. The main design challenge is to decide on a good approach direction for the hand. Given the two contact points, you can approach the grasp from any direction that is perpendicular to the line connecting the two contact points. Your job is to pick the most ergonomic approach direction. For instance, there may be some approach direction wherein if you placed the end-effector in that pose then it would be in collision with the object. You want to make sure this is not the case. Recall that you have access to a full mesh of the object. Use this to your advantage when doing collision checking. Look at figure 3 to get a better idea of what the gripper's reference frame looks like, and what the dimensions of the gripper are.

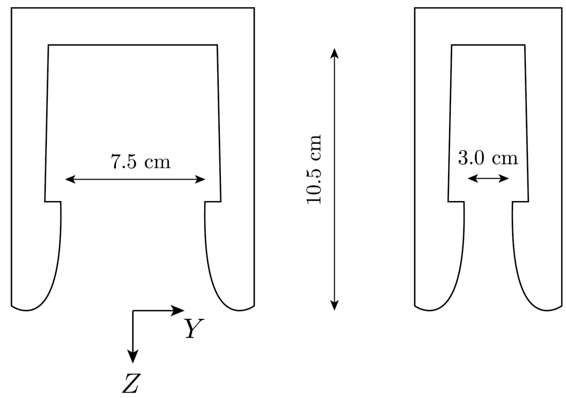


Figure 3: Dimensions and axes of the robot grippers. The X-axis points *into* the page.

3 Deliverables

To demonstrate that your implementation works, deliver a report with the following. The purpose of these project reports are to gradually scale up to a full conference-style paper, which you'll be writing for your final project. Please format your report using the IEEE two-column conference template. Column suggestions do not account for the figures.

1. An abstract, of at most 150 words, describing what you did and what results you achieved. Conveying information concisely is a difficult skill, and we want you to practice it here.
2. Methods: ($\sim$ 2-4 columns)
   (a) Describe your approach and methods in detail. Let us know of any difficulties you encountered and how you overcame them.
   (b) Describe your implementations of the grasp metrics. You do not need to repeat theory that is already present in the lab document, but you should mention any details necessary for understanding your implementation of the metrics. For instance, if you modified the optimization problem to explicitly incorporate the discretization of the friction cone, explain how you did that. Explain the software you are using as your optimization backend. Explain what parameters you chose for your algorithms and why.
   (c) A detailed description of your grasp planning algorithm.
3. Experimental results: Summarize your results in both words and with figures. The figures should all have descriptions so that they are understandable without needing to read the paper or context. ($\sim$ 1-2 columns)
   (a) For each object, score each of the five grasps using each one of the 3 metrics you implemented. Summarize your results in a table along with the success rate of each grasp, as described in the experimental data provided to you.
   (b) Include visualizations of 3 new grasps on each object that you sample and plan with using your custom planner.
4. Discussion: ($\sim$ 2-3 columns)
   (a) Discuss what difficulties you faced when designing your grasp planning algorithm. Were there any edge cases you did not anticipate?
   (b) Discuss any physical principles that appear to differentiate the successful grasps from the failures.
   (c) Discuss how predictive each of your grasp metrics were for the success or failure rate of the given grasps. Do the grasp metrics seem to have predictive power? Or do the grasps seem to succeed/fail in real life for other reasons?
   (d) Discuss the strengths and limitations of your custom grasp planning algorithm. Do you use any heuristics that depend on assumptions that may not always hold? Is your planner computationally efficient? Can you think of any failure cases for your planner?

5. A bibliography section citing any resources you used. This should include any resources you used from outside of this class to help you better understand the concepts needed for this project. Please use the IEEE citation format (https://pitt.libguides.com/citationhelp/ieee).

6. Page Limit: To prepare you for conference-style papers (and to protect the graders' time), reports are **limited to 6 pages**, not including the bibliography or appendix. While we are not enforcing a figure limit, we expect your figures to be concise, informative, and readable, with detailed captions for each.

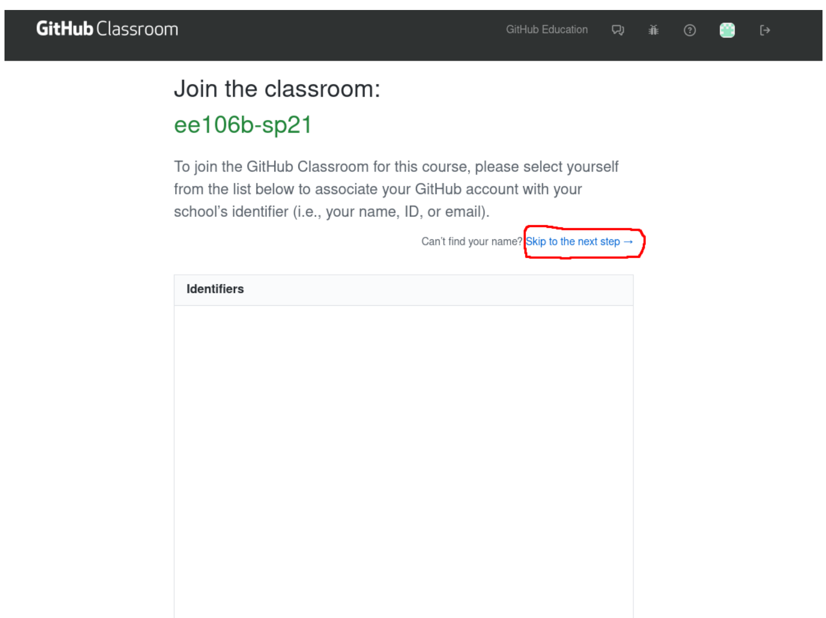
7. **Please list your github classroom team name in your report.**

8. **BONUS:** In addition to your report, write a couple paragraphs detailing how the project documentation and starter code could be improved with any comments you have regarding the pedagogical success of the assignment. What do you think were the pedagogical goals of this assignment? How well did it achieve these goals? How might it be made better so as to achieve them more effectively? This course is evolving quickly, and we're always looking for feedback. This should be a separate section in the report, and only well-considered, thoughtful feedback will merit full credit.

4 Getting Started

4.1 Github Classroom

For subsequent projects in this class, we will be using GitHub Classroom. To access the starter code, simply head over to the project 4 assignment page to accept the assignment. If you are asked to associate yourself with a school identifier, just click "Skip to the next step"



You should now be asked to create a team or join from a list of existing teams. If one of your teammates has already created a team, you should join that team instead of creating a new one. **After you have joined a team, you will not be able to switch teams by yourself. If you make a mistake or something else comes up that requires you to switch teams, let us know.** That's it! Your team should now have a private project repository located at https://github.com/ucb-ee106-classrooms/project-4-your-team-name. Let us know if anything went wrong.

4.2 Pulling starter code updates

If there are any updates to the starter code that you wish to pull you may do so with the commands

```
cd private-repo
git remote add public https://github.com/ucb-ee106/proj4_pkg.git
```

```
git pull public main
git push origin main
```

4.3 Conda Environment and Dependencies

If you don't already have it installed, please install Conda and create a Conda environment for this project. Instructions for installing Conda can be found here.
It is preferable to use Python 3.6 for this project. Create a new Conda environment by running the following command. This will also install Python 3.6 if you do not already have it.

```
conda create -q -n proj4_env python=3.6
```

Once you have created a new environment you can activate it by running

```
source activate proj4_env
```

Afterwards, please run the following commands

```
conda config --add channels conda-forge
conda install shapely rtree numpy scipy
```

Now you can `pip` install the necessary dependencies

```
pip install trimesh
pip install vedo
```

You can exit the conda environment at any time by running

```
source deactivate proj4_env
```

4.4 Starter Code

Videos of the pawn grasps can be found here. Videos of the nozzle grasps can be found here.

grasping.py This is the only file you will need to edit. All the functions you need to implement are in here, along with utility functions for loading in grasp data and visualizing the grasps.

pawn.npz A compressed numpy file format. The NPZ format saves what is effectively a dictionary of numpy arrays. See the `load_data` function in `grasping.py` to see how to load data from NPZ files. This file contains three arrays that together constitute data regarding the 5 grasps that we sampled and executed on the Pawn object:

1. `tip_poses` is an array that holds 5 rigid transforms describing the locations of the gripper tip for each of the grasps, in the reference frame of the object.
2. `grasp_vertices` is an array that holds 5 pairs of size 3 arrays, where each pair is the (x, y, z) location of the two points of contact of the grasp, in the object's reference frame. These were sampled from the mesh of the object.
3. `normals` is an array that holds 5 pairs of size 3 arrays. For each grasp, this array stores the normal vector to the mesh at the two contact points. Note that the normal vectors to a closed mesh always point OUTWARD from the mesh.
4. `results` is an array storing the results of each of the five trials for each of the five grasps. This is a 5x5 array, where the (i, j) entry holds the result (1 for success, 0 for failure) of the jth trial for the ith grasp.

nozzle.npz Same as above, but for the Nozzle object.

pawn.obj A triangular mesh for the Pawn object, specified as an OBJ file.

nozzle.obj A triangular mesh for the Nozzle object, specified as an OBJ file.

utils.py A set of utility functions that you may or may not find useful.

5 Scoring

Table 1: Point Allocation for Project 4.

Section	Points
Code (includes listing your Github team name)	5
Page Limit	10
Figures: Quality and Readability	10
Abstract	5
Methods	15
Results: Gravity Resistance Metric	5
Results: Ferrari Canny Metric	5
Results: Robust Force Closure Metric	5
Custom Grasp Planner	15
Discussion of Results	15
Bonus: Future Improvements	5*

See table 1 for the full point breakdown. This project will be out of 90 points with an additional 5 points possible.

6 Submission

For this project you will submit your project report on Gradescope and your code to Github Classroom. Simply push your code to the private repository that was created for your team. We will be able to see any changes you push to your assignment repository. Please add your groupmates as collaborators on both Gradescope and Github.
Please remember to list your Github classroom team name in your report.

7 References

[1] R. M. Murray, S. S. Sastry, and L. Zexiang. *A Mathematical Introduction to Robotic Manipulation*. 1st. USA: CRC Press, Inc., 1994.

[2] C. Ferrari and J. F. Canny. "Planning optimal grasps."