EECS C106B: Project 4 - Grasp Planning with Sawyer [*]

Due date: March 22nd at 11:59pm

Goal

The purpose of this project is to combine many of the topics presented in this course to plan and execute a grasp with a manipulator in the lab. This will consist of detecting the object to grasp, planning a stable grasp, moving into position, closing the grippers, and lifting.

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. 2020 and 2019 versions reworked into 2022 version by Riddhi Bagadiaa and Jay Monga. Adapted for Spring 2024 by Kirthi Kumar

1 Theory

Here's a quick refresher on grasp theory drawn from MLS. 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:

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

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 as the following:

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

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

$$G_i := Ad^{\top}_{g^{-1}_{oc_i}} B_{c_i}.$$

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

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

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, the 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

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

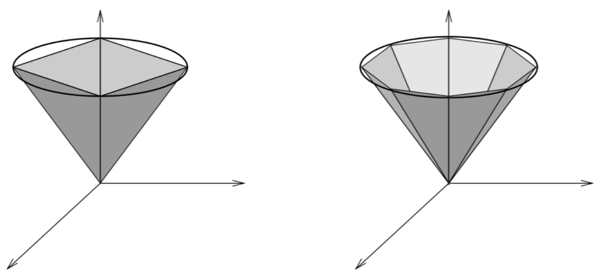

Figure 1: Approximations of the friction cone. From section 5.3 of MLS

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.
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 (whether or not it's a force closure) 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 the following:

$$\begin{aligned} \widehat{f} =_f & \|f\|_2^2, \\ \text{s.t.} \quad & f \in FC, \\ & Gf = -W_g, \end{aligned}$$

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

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

where $\widehat{f}_{3_{c_i}}$ is the perpendicular finger force at contact i, and is also denoted as $\widehat{f}_{c_i}^{\perp}$. 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`).

1.2.2 Ferrari-Canny

The Ferrari-Canny metric 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

$$\begin{aligned} LQ(w) &= \max \frac{\|w\|}{\|f\|}, \\ \text{s.t.} \quad & f \in FC, \\ & Gf = w. \end{aligned}$$

$LQ(w)$ 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

$$\begin{aligned} LQ(\hat{w}) &= \max \frac{1}{\|f\|}, \\ \text{s.t.} \quad & f \in FC, \\ & Gf = \hat{w}. \end{aligned}$$

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

$$\begin{aligned} \widehat{LQ}(\hat{w}) &= \min \|f\|_2^2, \\ \text{s.t.} \quad & f \in FC, \\ & Gf = \hat{w}. \end{aligned}$$

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 the following:

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

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 $\{\hat{w}_i\}_{i=1}^N$ randomly from the unit sphere, finding $\widehat{LQ}(\hat{w}_i)$ for each $\hat{w}_i$, 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 $F : SE(3) \times M \to \{0, 1\}$ be the proposition:

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

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 (μ, M) will be the following:

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

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 $\delta_x, \delta_y \sim \mathcal{N}(0, I\sigma^2)$ (i.i.d.) 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

$$\hat{\mu} = \mu \cdot e^{\hat{\delta}}.$$

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 Getting Started

Create a new folder in your ROS workspace to store your code for this project. Remember to build and source your workspace after you download the starter code. If you're switching your team, make sure to fill out the team form.

2.1 Github Classroom

You can use a repo from this and add your teammates here: https://classroom.github.com/a/53uRCrYS. **If you already made one in Project 3, you may use the SAME repo for Project 4 (starter code for both projects should be accessible through this link).**

2.2 Configuration and Workspace Setup

The packages you need for this project should already be installed in your workstation. However, if you're running into any issues, you can run pip install to locally install anything you need.
You will also need to download the AR tracking package into your workspace. Please refer to Project 1 for AR Tracking/camera setup instructions, including enabling the robot, working with MoveIt!, etc.

2.3 Starter Code

Your stater code is pretty bare bones – it only includes at `utils.py`, `grasping.py`, and the objects and grasping data. Definitely take a look at `utils.py`. In general, you should use MoveIt! to control the robot, and set up a publisher and/or subscriber to get and use the pose of the end effector. A good theory refresher can be found in Chapter 5 of MLS. (By the way, feel free to change anything you'd like to in the starter code.)

3 Project Tasks

3.1 Part 1 - Simulation

Your starter repo contains some grasp data (in .npz format) and some objects (in .obj format). Your task will be to translate the theory above into grasping.py. This should allow you to check your approach before you move on to the hardware. The grasping.py is pretty self-contained (alongside the referenced theory above), so please take a peek at the starter code. You may get what seems like an error (something along the lines of only 2 data points found or similar) once you've exited out of the images that should pop up, don't worry about it.

3.2 Part 2 - Hardware

On the hardware, you'll try implementing this grasping theory for the Sawyer. The first place you should start is Lab 5 from EECS 106A, which is linked for your convenience here. You can use this as your starting point, but you have to modify it to accommodate for the specifics of this project. This part of the lab is really open-ended – how exactly you want to implement it is up to you. You can also take a look at Project 7 from 106A and your Project 1 for inspiration on how to implement this.

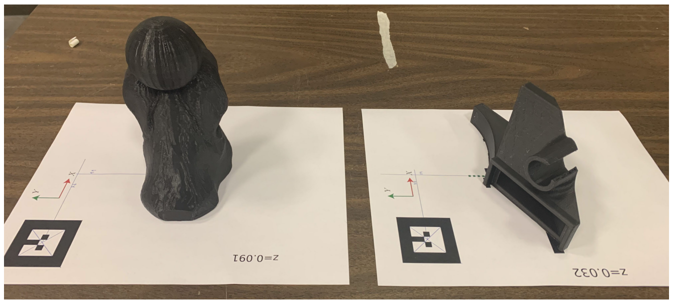

Figure 2: Inspo for calibration (athough you could implement it however you'd like)

As illustrated above, one challenge you're going to have is with the calibration of the environment. You're encouraged to place the AR tag a fixed distance away from the object, and you can offset the location of the pickup through that. However, you're welcome to try any calibration method as long as it works.
You should try a soft finger contact model with $\mu = 0.5$ and $\gamma = 0.1$. For this project, we ask that you plan three grasps for two different objects (the nozzle and the pawn from sim – they've been 3D printed and placed in a box in the front of the room). You have to test all three grasp metrics: 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. **So, in total, you'll have 18**

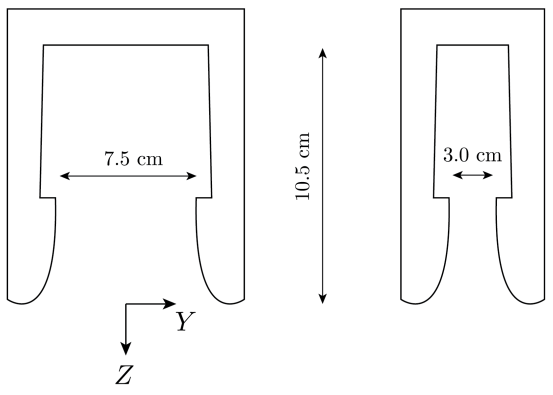

Figure 3: Approximate gripper dimensions (these might be slightly different for your robot, so please double check them if you can).

grasps!

The general workflow for planning an executing a grasp is as follows:

- Sample candidate point grasps from a 3D mesh model of each object by taking pairs of the vertices.
- Compute the quality metric for each grasp.
- Compute the gripper pose relative to the object for each grasp.
- Choose a set of three gripper poses based on their grasp metric score. Make sure the gripper will not collide with the object.
- Determine the hand pose in the robot's frame of reference using AR markers.
- Control the Sawyer arm and move the hand into the proper position.
- Close the grippers.
- Lift the object and determine whether or not the grasp was successful.

Now, measure the (binary) success rates of each grasp according to 1) being able to lift the object and 2) being able to lift the object and place it down in another location with the object toppling. You should visualize this data in your report.

We understand that this part of the lab is hard/not going to always work. What matters in this part is more of the attempt made to pick up the objects, rather than the actual accuracy of the ability to pick up the object. But, if you have some time, you're encourage to play around with the different parameters (and document that in your lab report).

3.2.1 Extra Credit - Vision

For up to 10 points of extra credit, do the above for any object in the lab (a cube is recommended, as it is a lot easier than most other objects in the lab). To do this, you will need to use vision to determine the object's size and pose without use of an AR tag (or some combination of vision + an AR tag). Once these parameters have been determined, you can feed the object's known geometry (a lot easier if it's a cube) into the rest of your grasping pipeline.

4 Deliverables (Lab Report)

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. Abstract: 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: Describe your approach and methods for implementing the workflow described in Section 3.2 Part 2 (Hardware). Let us know of any difficulties you encountered and how you overcame them. (2-4 columns)
    (a) Extra Credit: If you implemented this, specifically describe your procedure for reconstructing the size and pose of the cube/object.
    (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.
    (d) For each of 18 grasps, include images of the grasp point visualization outputted by your code and a video of the robot attempting your grasp. All videos should be stitched together into a single video no longer than 2 minutes with each grasp labeled and speed-up factor included.
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. (around 1-2 columns)
    (a) Extra Credit: If you implemented this, for the object, compare the size and pose computed by your computer vision algorithm to the actual size and pose of the cube. How accurate (qualitatively and quantitatively) is your algorithm? You may use an AR tag to help measure accuracy of your algorithm.
    (b) For each object, record the scores of each of your planned grasps for each of the three different grasp metrics (15 scores total per object). Summarize your results in a table along with the success rate of each grasp (success or fail for the object).
    (c) Summarize the results for your grasping pipeline.
4. Discussion: (around 2-3 columns)
    (a) Discuss how performance differed across the objects. Which object was the hardest? Which was the easiest? Why?
    (b) Discuss what difficulties you faced when designing your grasp planning algorithm. Were there any edge cases you did not anticipate?
    (c) 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?
    (d) Please include a specific subsection dedicated to the sim to real challenges, addressing (but not limited to) the items below:
        i. Discuss any physical principles that appear to differentiate the successful grasps from the failures.
        ii. Discuss how predictive each of your grasp metrics were for the success or failure rate of the given grasps.
        iii. Do the grasp metrics seem to have predictive power? Or do the grasps seem to succeed/fail in real life for other reasons?
    (e) Discuss what worked well for the grasping pipeline, and what could be improved.
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.
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. Appendix:

   (a) GitHub Link: Provide the link to your GitHub classroom repo. Simply push your code to the private repository that was created for your team, and we will be able to see any changes you push to your assignment repository.

   (b) Videos of your implementation working. Provide a link to your video in your report.

   (c) (Optional) How can the lab documentation and starter code be improved for the future?

5 Scoring

Table 1: Point Allocation for Project 4

Section	**Points**
Page Limit	5
Abstract	5
Methods	10
Results: Gravity Resistance Metric	10
Results: Ferrari Canny Metric	10
Results: Robust Force Closure Metric	15
Discussion	10
Sim to Real Discussion Subsection	10
Bibliography	4
Code	3
Video	3
Bonus*: Cube/Object Recognition Task	10
Total	85

With bonus points, you may be able to earn up to 95 out of 85 points. Good luck!