Assignment 1

Assignment Objective

Benchmark Python coding skills. Python is an important programming language and very prevalent in robotics. It is also freely available, cross-platform, and great for rapid prototyping. Code developed with Python is accessible to anyone with a computer, making it a great choice for development and deployment. This is contrast to languages such as MATLAB that require licences not easily available to everyone. We will use the Numpy library primarily for numerical computations. This assignment will introduce you to some basic functionality of this library. If you are new to numpy, this Numpy Tutorial is a great introduction. Also, we recommend you to take a look at the post Look Ma, No For-Loops: Array Programming With Numpy which will help you avoid unnecessary `for` loops and really boost your code.

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 Canvas. 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_1.py` from the Files menu of Canvas.
- For each "Part" of the HW assignment, fill in the missing code as indicated by the comments.
- Submit the completed code to Canvas.
- Each part has an equal contribution to your score and the total is equal to 100.
- The assignment is due on September 20^{th}, 2021.
- You will need the `numpy` library.

Part 1

Consider the polyhedral cone depicted in Fig. 1. The cone is constructed by N edges defined as:

$$\boldsymbol{v}_i = \begin{bmatrix} \cos\theta\cos\phi \\ \sin\theta\cos\phi \\ \sin\phi \end{bmatrix}$$

where $\theta = (2\pi i)/N$ for $i = 0, \cdots, N-1$. The cone defined by these edges is written as:

$$\mathbf{V} = \begin{bmatrix} \boldsymbol{v}_0 & \cdots & \boldsymbol{v}_{N-1} \end{bmatrix}$$

The polyhedral cone is a common approximation to the Coulomb friction cone (covered in the course notes). Complete the function `generate_edges` in `assignment_1.py`. This function generates a cone given its parameters:

$$\mathbf{V} = \text{generate\_edges}(N, \phi)$$

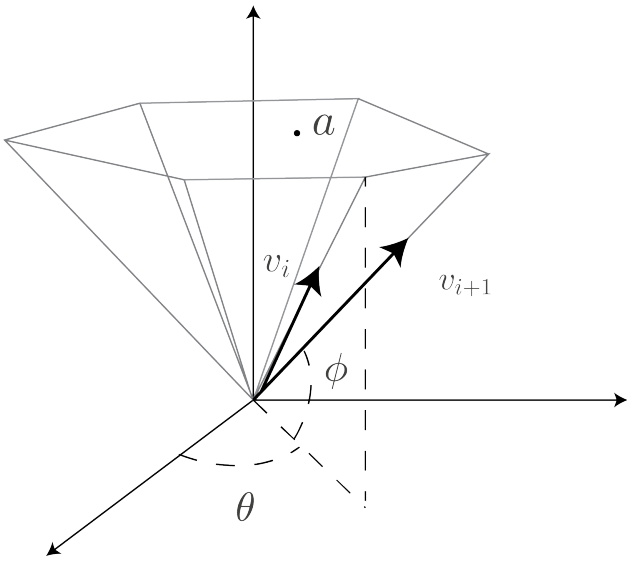

Figure 1. Polyhedral cone parameterized by ϕ and N.

Part 2

We can compute the facet normals (pointing inwards of the cone) using the cross-product of the edge vectors:

$$\boldsymbol{s}_i = \boldsymbol{v}_i \times \boldsymbol{v}_{i+1}$$

The cone generated by facet normals can be written as:

$$\mathbf{S} = \begin{bmatrix} \boldsymbol{s}_0 & \cdots & \boldsymbol{s}_{N-1} \end{bmatrix}$$

Complete the function `compute_normals` in `assignment_1.py`. This function generates the facet normal cone given the edge cone:

$$\mathbf{S} = \text{compute\_normals}(\mathbf{V})$$

Part 3

We can compute the distance of any point $\boldsymbol{a} = (a_x, a_y, a_z)$ in the interior of the polyhedral cone to facet i using:

$$d = \frac{\boldsymbol{s}_i \cdot \boldsymbol{a}}{||\boldsymbol{s}_i||}$$

Complete the function `compute_minimum_distance_from_facet_normals` in `assignment_1.py`. This function computess the minimum distance d^* of a given point $\boldsymbol{a}$ to the facet normal cone:

$$d^* = \text{compute\_minimum\_distance\_from\_facet\_normals}(a, \mathbf{S})$$

Part 4

We're going to string together the functions above. Complete the function `compute_minimum_distance.py` in `assignment_1.py` that computes the minimum distance d^* of a given point $\boldsymbol{a}$ to a polyhedral cone parameterized by N and ϕ:

$$d^* = \text{compute\_minimum\_distance}(\boldsymbol{a}, N, \phi)$$

Using the functions previously written for this purpose.

Part 5

Complete the function `check_is_interior_point` in `assignment_1.py` to check if a given point $\boldsymbol{a}$ is in the interior of a polyhedral cone parameterized by N and ϕ. The function should return true if the point is inside the cone, and false if it is outside. Hint: you can use the functions you wrote for parts 3 and 4. Think about what it means for a point to be in the interior of a polyhedral cone in terms of distances.