Robotics - Lab 2

Taiming YuenJames

January 30, 2026

1 Link and Arm Lengths

Based on the xacro file provided, the link lengths add up to a total reach of approximately 1.0872 meters.

$$0.425 + 0.39225 + 0.093 + 0.09465 + 0.0823 = 1.0872$$

2 Maximum Range

In the x direction, I am able to achieve a maximum range of 0.948992457 meters. In the y direction, I am able to achieve a maximum range of 0.948989495 meters. In the z direction, I am able to achieve a maximum range of 1.03122057 meters. I am not able to reach the theoretical maximum range exactly because the points I chose (e.g. [0, 0, 2]) are optimally solved diagonally (with respect to the base), yeilding a shorter maximum length.

Code

```
import ikpy.chain
import numpy as np
import math
from ikpy.utils import geometry
from scipy.spatial import ConvexHull
import ikpy.utils.plot as plot_utils
import warnings
warnings.filterwarnings("ignore", category=UserWarning)

# Question 3: generate 3D volume of reachable points
def sphere():
        my_chain = ikpy.chain.Chain.from_urdf_file("./ur5/
           ur5_gripper.urdf") # import arm
        points = []
        num_points = 1000 # reach for this number of points
        for i in range(num_points):
            gauss = np.random.normal(0, 1, 3) # generate random
               x, y, z values with normal distribution
            target_position = gauss / math.sqrt(gauss[0] ** 2 +
               gauss[1] ** 2 + gauss[2] ** 2) # calculate target
                positions on a sphere
            solved_position = my_chain.forward_kinematics(
               my_chain.inverse_kinematics(target_position)) #
               calculate solved position
            points.append(solved_position[:3, 3]) # save
               position of end effector
        fig, ax = plot_utils.init_3d_figure()
        points = np.array(points)
        hull = ConvexHull(points)
        ax.plot_trisurf(points[:, 0], points[:, 1], points[:, 2],
           triangles=hull.simplices, color='cyan', alpha=0.3,
           edgecolor='k') # generate convex hull
        ax.set_box_aspect([1, 1, 1])
        plot_utils.show_figure()

# Question 4: collision checking
def collision(joints):
        my_chain = ikpy.chain.Chain.from_urdf_file("./ur5/
           ur5_gripper.urdf") # import arm
        solved_position = my_chain.forward_kinematics(joints,
           full_kinematics=True) # calculate solved position
        for index, link in enumerate(my_chain.links): # get position
            of each wrist
                translation, rotation = geometry.
                   from_transformation_matrix(solved_position[index
                   ]) # save wrist position
                z = translation[2] # save wrist z position
                if z <= -0.01: # if intersecting with XY plane, fail
                        print("FAIL:␣Arm␣at␣current␣joint␣angles␣
                           intersects␣with␣the␣defined␣plane")
                        break
        else:
                print("PASS:␣Arm␣at␣current␣joint␣angles␣does␣not␣
                   intersect␣with␣the␣defined␣plane")

        fig, ax = plot_utils.init_3d_figure()
        xlim = ax.get_xlim()
        ylim = ax.get_ylim()
        xx, yy = np.meshgrid(np.linspace(*xlim, 10), np.linspace(*
           ylim, 10))
        zz = -0.01 * np.ones_like(xx)
        ax.plot_surface(xx, yy, zz, color='red', alpha=0.2) # plot
           XY plane (translated to account for arm base)
        my_chain.plot(joints, ax) # plot arm
        ax.set_box_aspect([1, 1, 1])
        plot_utils.show_figure()

if __name__ == "__main__":
    sphere()
    num_test_cases = 5 # set desired number of test cases
    for i in range(num_test_cases):
        joints = np.random.uniform(-1, 1, 8) # generate random joint
            angles
        collision(joints)
```

Visualizations

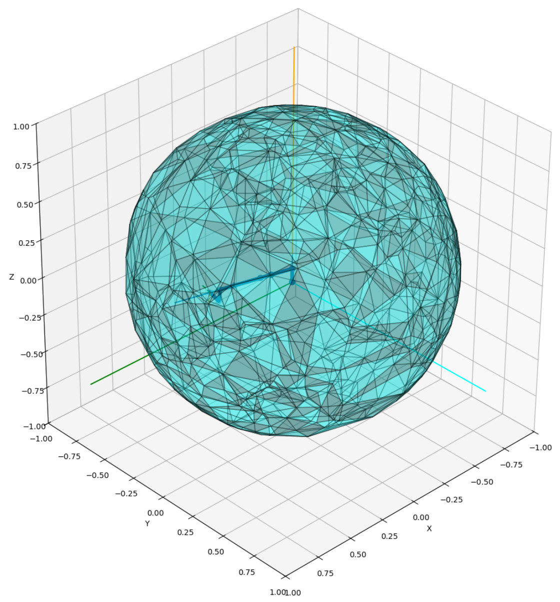

Figure 1: Reachable Poses as Convex Hull

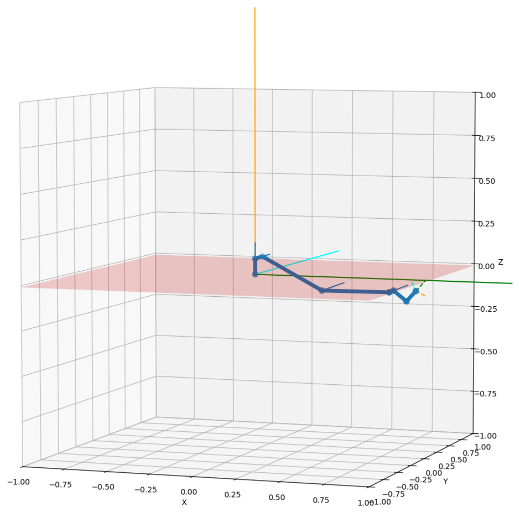

Figure 2: Collision Checking Test Case Failure

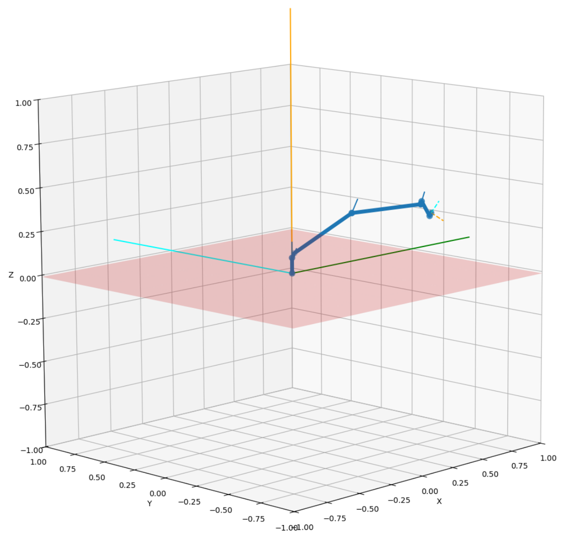

Figure 3: Collision Checking Test Case Pass