Lab 4: Planning in Task-Space Regions

In the last lab, you implemented an RRT that generated a motion plan from a start to a goal configuration to grasp an object. However, the robot typically does not know the goal configuration in advance. Instead, it knows only the position of the object in world coordinates, as extracted for instance from sensing. In this lab, you will implement a planning pipeline that takes as input the 3D position and identity of the can, reaches the object and pick it up. During development, you will need to run your code in simulation with

```
$ python soda_grasp_ik.py --sim
```

1. We will describe the grasping task with a Task Space Region (TSR) constraint. Read [1] through Section 4 to build an understanding of TSRs.

2. Implement the TSR constraint for the can. Specify the bounds matrix $\mathbf{B}^w$, so that TSRs are sampled uniformly around the can at the same height. You can do this by filling in the functions `createBw`, which will be called by `createSodaTSR`. Use the following values for $\mathbf{B}^w$:

$$
\begin{aligned}
x_{min} &= x_{max} = 0 \\
y_{min} &= y_{max} = 0 \\
z_{min} &= -0.02,\ z_{max} = 0.02 \\
\psi_{min} &= \psi_{max} = 0 \\
\theta_{min} &= \theta_{max} = 0 \\
\phi_{min} &= -\pi,\ \phi_{max} = \pi
\end{aligned}
$$

   You can visualize the TSRs in **rviz** by calling the function `viewer.add_tsr_marker(sodaTSR)`. You can use the visualization of the end-effector frame as reference. Set the height so that the robot can grasp the can without colliding with the table. **Save a picture** of your simulation in rviz as `tsr_vis_1.png`

3. Modify the $\mathbf{B}^w$ matrix, so that only TSRs in the semicircle facing the robot are sampled (the robot will not have to go behind the can to grasp it). Visualize the result. **Save a picture** of your simulation in rviz as `tsr_vis_2.png`

4. The pipeline then calls `ik_generator` to compute IK solutions for the robot, using the TSR constraints that you specified. For each computed IK configuration, **use the RRT planner that you have implemented in Lab 3** to find a motion plan. Once the RRT planner finds a plan for a configuration, execute and visualize the trajectory.

5. Now the robot should grasp the object. To do that, call the function `close_hand(hand, preshape)`, which takes as input the hand. The preshape is a vector (f_1, f_2), where $0 \leq f_1, f_2 \leq 1.6$. Specify a value for the fingers that encompass the can without grasping it too tightly. Then, call the function `hand.grab(soda)` which will attach the object to the robot's end-effector. Execute and visualize this procedure.

6. Lift the object vertically 0.5 m using the Jacobian pseudoinverse method. You can call the following function to get the Jacobian frame:
   `arm_skeleton.get_jacobian(hand.get_endeffector_body_node())`

   The first 3 columns of the Jacobian matrix are the rotational portion, and the next 3 the linear (translational) portion. Then, compute the pseudoinverse using `numpy.linalg.pinv` function. Use the function `arm_skeleton.get_positions()` to get the configuration of the arm, and `ada.set_positions(q)` to set the configuration of the arm to q. Iterate using a vertical displacement 0.01 until you have lifted the arm 0.5 m. Using the `ada.set_positions()` function to show the motion of the arm, visualize the result.

7. This time do only one iteration of the Jacobian pseudoinverse with vertical displacement 0.5 m. Record your observations and justifications in `answers.pdf`.

8. Execute the whole pipeline and **record the result**. Save the video as `full_trajectory.mp4`.

Extra Credit: Once you are confident in your simulation results, you are ready to run it on the real robot. This can be done on the lab workstations with

```
$ python soda_grasp_ik.py --real
```

Refer to Piazza for more instructions on scheduling time in the lab.

References

[1] Berenson, D., Srinivasa, S., and Kuffner, J. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research 30*, 12 (2011), 1435–1460.