Andrew Valle

John Rogers

CSCI445 Final Project Write-up

Getting the Mobile Robot to the Arm

1. First, since the RRT Algorithm as the Particle Filter is based off of some aspect of randomness, A random seed was used in order to achieve the same results every time the program runs. This was important because using the RRT as an example, the robot may have taken a slightly different path to its goal location each time the algorithm runs.
2. Next, a goal location is needed to provide for the RRT algorithm. A function was created which takes the location of the arm (simulation x-y coordinates in meters) and converts the coordinates to the relative position in pixel coordinates of the configuration space map. Depending if the arm is on the north, south, east, or west, we use the same x or y coordinate, plus or minus some offset which relates to the angle of the arm when going down. The code can be seen here:

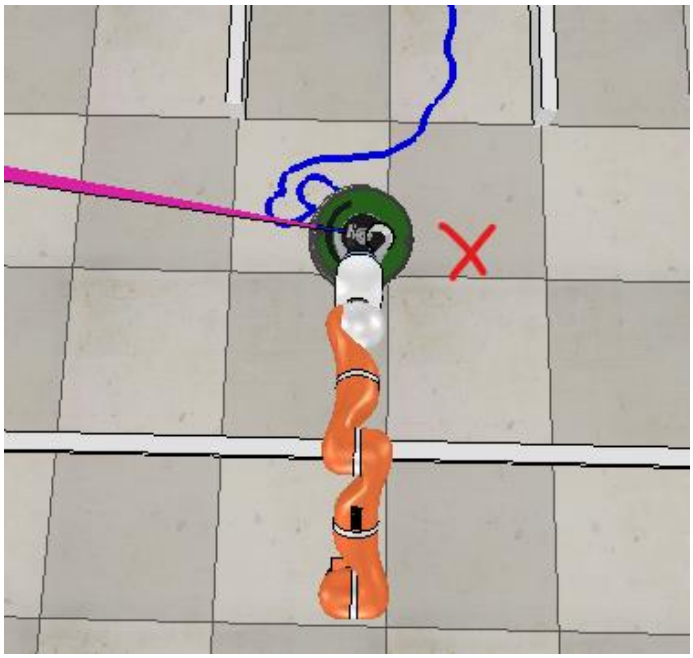
```
def get_arm_position_in_pixels(self, position):
    # If arm on East or West
    if 0.0250 <= position[1] <= 3.0250:
        if position[0] < 0.0250:
            p = (((position[0] + 0.96) * 100), self.map.height - ((position[1]) * 100))
            return p
        else:
            p = (((position[0] - 0.96) * 100), self.map.height - ((position[1]) * 100))
            return p

    # If Arm is on North
    elif position[1] > 0.0250:
        p = (((position[0])*100), self.map.height - ((position[1] - 0.96) * 100))
        return p

    # If Arm is on South
    else:
        p = (((position[0])*100), self.map.height - ((position[1] + 0.96) * 100))
        return p
```

3. After getting the position of the arm in the relative pixel coordinates, we know our current position of the mobile robot by using the function called sim_get_position(). We use that as well to provide it to the RRT as the start position.

4. Now that the RRT has both a start and goal location, we then use that start location to start off all the particles in the particle filter to begin at the start position of the robot. Here we are considered localized since we currently know where we at.
5. Now that the key elements are initialized, it is time to generate the path to the arm by using the RRT algorithm.
6. Once generated, to mobile robot continues to follow the path.
7. Every few waypoints in the path, we check the robot's localization status. If our estimated position is too far off from the robot's actual location, we start localizing again by simply rotating in place while taking measurements until localization is complete. After the robot is done with the localization process, it continues following the path to the arm.
8. Repeated step 7 until finally reaching the goal.
9. Once the goal is reached, there were more things to consider:



This is what happens once the goal is reached. This is not the behavior that we want. The goal is for the arm to pick up the cup. Therefore, we added for the robot to rotate and face its back side towards the arm and then move slightly forward. Adjustments were changed depending on the location of the arm whether it is on the north, south,

east, or west. Once these changes were made, the arm was able to successfully pick up the cup.

Picking Up the Cup and Placing it on the Shelf

The first step in creating the behavior of the robotic arm was to establish a point within its workspace that was accessible to the mobile robot. With respect to the robot arm in default position, the position $0.98\ m$ directly in front of the robot, at a height of $0.17\ m$ above the robot arm's base was picked to be the rendezvous point between the end effector of the arm and the mobile robot. Commands to the arm were in coordinates with the arm base as the origin. This ensures that the arm extends out in the same fashion in reference to itself, independent of how it is positioned within the environment. The coordinates of the rendezvous point are converted from the arm base coordinate system to the simulation environment coordinate system, which are then communicated to the mobile robot as its goal location.

The height of each shelf was obtained by iteration. It was assumed that the placement of the rack in relation to the robot arm would be constant, independent of the configuration of the

environment. All motion by the robotic arm is essentially predetermined. The only input required by the user is the desired shelf to place the cup.

Motion within the y-z plane was induced by inverse kinematic principles. Motion was constrained to be available for only two joints, 2 and 6. This ensured a simple solution to the inverse kinematic equations. This process allowed for placement of the end effector in order to grab the cup and place the cup on the shelf, with no z axis rotation taking place. Rotation about the z axis was executed in a fixed maneuver. The only consideration taken while rotating the robot arm in order to face the shelf was ensuring enough clearance to pass over the maze wall. When placing the cup either on the lowest shelf or the highest shelf, joint 6 on the robotic arm had to be angled down to ensure clearance of obstacles and proper release of the cup.

Lastly, once the cup was within the grip of the end effector, consideration had to be taken to ensure that the momentum of the system would not cause the cup to be released by the gripper. Instead of having the robot arm move directly from one position to another, the motion was implemented incrementally by degrees enough to not produce too much jerk.