ECE 590.04 Homework 2: Transformations and Kinematics

Due date: 2/4, 4:40pm

In this assignment you will use the Klamp't interface to calculate certain key elements of the Amazon Picking Challenge.

GUI Overview. If you run "python hw2.py", you will see a visualization of the APC shelf, some boxes, and the Baxter. The red wire box shows the order bin, and the orange wire boxes show the objects. Once an object is perceived, it is drawn as a solid orange box. The camera coordinate frames and the gripper coordinate frames are also drawn (red: x, green: y, blue: z).

The robot's main control loop is defined in methods of the PickingController class, which has three major high-level actions.

- viewBinAction(b). Move one of the cameras in the robot's wrist to observe a bin. This will invoke a fake perception module that will estimate the poses of all objects in the bin.
- graspAction(). Move the gripper to grasp an object in the current bin.
- placeInOrderBinAction(). Move the grasped object to the order bin.
- fulfillOrderAction(order). Given a list of objects, find all of them and place them in the order bin.

In the GUI, you can control these actions using the keyboard:

- Keys 'a'-'l' request a viewBinAction()
- 'x' requests a graspAction()
- 'p' requests a placeInOrderBinAction()
- 'o' requests a fulfillOrderAction(order) for a given order

It is your job to implement some key portions of these actions.

Modeling. Bins are labeled A-L, and the objects need to be sensed, grasped, and placed into the order bin. In this assignment we assume the robot can jump between configurations. We ignore collisions, and assume the robot can grasp objects instantaneously as long as the gripper is in the right location.

The camera frame has the camera pointing forward along the z axis. The gripper frame has the gripper pointing in the z direction and closing and opening along the x axis. There is a notion of the "center" of the gripper, which is where the forces of the fingers are applied. It is given by an offset X_gripper_center_xform from the local gripper frame, where X is either left or right.

The shelf frame is defined at the bottom center of the shelf, and has a coordinate system that is rather odd. The x direction increases from left to right, y increases from bottom to top, and z increases from back to front. The robot's estimate of the shelf frame in world coordinates is found in RobotController.knowledge.shelf_xform (for this assignment it is known exactly).

The bin locations and dimensions are defined by the apc.bin_bounds object.

Initially, the item locations and bins are not known. The fake perception module produces a list of all items and their poses whenever a sensing operation is performed.

API Documentation. Your code will mainly be added into hw2.py, which contains some skeleton code for the overall robot control loop as well as the visualization code. Several constants regarding the APC bin and items are also found in apc.py. You will need to understand some items in this file as well.

It is your job to fill in the move_camera_to_bin(),move_to_grasp_object(),move_to_order_bin(), and fulfillOrderAction() methods of the PickingController class.

The KnowledgeBase class contains dynamic knowledge about perceived objects.

The Klamp't Python API is documented in greater detail in http://motion.pratt.duke.edu/klampt/pyklampt_docs. In particular, for this assignment you will need to spend some time understanding the modules so3.py, se3.py, robotsim.RobotModel, and ik.py.

Questions.

1. Implement the move_camera_to_bin() method. This method should choose an arm in PickingController.active_limb and calculate a configuration PickingController.config that moves a camera to a vantage point that looks at the designated bin from 20cm away from the open face of the bin.

   Use the objective() and solve() methods in the klampt.ik module to calculate inverse kinematics solutions. Note that you may need to define suitable initial configurations so that the local optimizations in the ik module can successfully find a solution.

   Do not worry about collisions for the purposes of this assignment, your solutions are allowed to collide but not violate joint limits (joint limits will be automatically handled by the ik module).

   You can test your method by pressing all the keys from 'a'-'l' in the GUI and verifying that the robot's camera frame is successfully moved to a suitable vantage point.

   [To help understand the global variables and operations on rigid transformations in in se3.py, it is recommended that you set the MyGLViewer.draw_bins to True and implement the KnowledgeBase.bin_vantage_point method. If your implementation is successful, the GUI will draw light green camera vantage points for each bin.]

2. Implement the move_to_grasp_object() method to choose a grasp, arm, an configuration for PickingController.config that moves the gripper to a suitable location on the object. The gripper must be moved so it touches one of the object's *candidate grasp frames* in object.info.grasps.

[It is recommended to visually debug your methods by implementing the KnowledgeBase.grasp_xforms method, and set MyGLViewer.draw_grasps to True to draw the candidate grasp frames around the objects. When you grasp an object your method should produce a configuration such that the gripper frame exactly lines up with one of the candidate grasp frames.]

3. Implement the move_to_order_bin() method to move a grasped object to a location over the order bin.

4. Implement the fulfillOrderAction () method. This method should automate the process of sequencing viewBinAction(),graspAction(), and placeInOrderBinAction() calls so that all of the requested objects are successfully placed in the bin if possible. The order of placing the requests into the order bin is unimportant, but the total number in the order does matter (e.g, an order of ['tall_item','tall_item'] requests two 'tall_item's). The method should return False if the order cannot be fulfilled.
   A priori, you should not assume that the identities of objects in each bins are known -- this is slightly different from the real APC where you will know which bins contain which items. Specifically, your method should work regardless of whether the objects on the shelf are changed, or if the default order is changed.

Submission. Submit your hw2.py code and any other files that you have changed or added and submit them on the Sakai website.