ECET 581 - Programming Robots with ROS

Lab 3 – Due Oct 12, 2017 – v1.1

This lab can be done in your groups of 2 or 3 people.

1. Use Baxter's inverse kinematics (IK) and, if necessary `tf2_ros` (transform) packages to do the same menu-based jogging of movement you did in lab 1, but this time, "jog" the motion **along the unit axes (X,Y,Z,R,P,Y)**. Use the same key strokes (1-6 and shift-1 – shift-6 for the left arm and q-y and Q-Y for the right) to send increments that are interpreted as "jogs" along the axes by the receiver. The receiver will use inverse kinematics to convert the updated goal locations in the base frame to joint angles. Also, print, to the screen, the resulting pose (position and orientation) of the wrist in the Baxter's base frame.

Set up a time to demonstrate your menu-based programs and upload the files to Lab 3. Turn in a screenshot of your program displaying the pose in two rather different configurations.

2. In your groups of about three, you will now create a visual servoing application to use SSD (sum of squared differences) to move the real Baxter robot to track a moving object. (You remember how to connect to the robot, right?) Hmmm. This will require you to set up frames of reference that you can transform between, then send appropriate commands of movement measured in the camera's frame transformed to the robot arm's frame. Start with one wrist camera looking down. Find an object with a small distinct pattern that is not easily confused (a "fiducial" or "feature" similar to the "good" features examined in Lab 2). Then, move the robot continuously in X and Y to put the fiducial (image patch or feature) in the center of the camera's field of view. (You can try this in the Gazebo simulator, but will need to create an interface to insert a block and move it with key strokes, maybe?) Also, you are recommended to use OpenCV, if you are familiar with that package (import "CameraController" and "cv2"). It is distinct from ROS, but ROS has adopted it as the recommended (open source) computer vision method. (Use the "image_view" tool to see the images from the cameras.)

   For each new image, record the time, the (x,y) location of the robot and the errors in x and y of the image feature from the center of the image.

   We are simplifying the problem by tracking in a plane parallel to object motion and allowing clean backgrounds with clear image features with slow motions. Since you are not moving in Z, you can choose a fixed-size image template for the feature (or image patch) that is appropriate for the object you've chosen to track.

   See http://sdk.rethinkrobotics.com/wiki/Camera_Control_Tool

Set up a time to demonstrate your visual servoing program and upload the files to Lab 3 (581-013). Turn in a PDF report with three sequential pictures looking down on the wrist and object while it tracks in x and y, an x-y plot of the robot's motion during tracking, and plots versus time of x and y errors.

Updated: Oct 4, 2017