ECET 581 - Programming Robots with ROS

Lab 1b – Due Sept 13, 2019 – v1.1

This lab can be done in your groups of 2 or 3 people. We will create a program that tracks an object in the camera and moves the robot so that the object stays in the center of the camera frame. Expect to figure some things out on your own.

4. 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 1a, but this time, "jog" the motion (make small frame increments per key press) **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 1, problem 4, on BlackBoard. Turn in a screenshot of your program displaying the pose (you should know what this means) in two rather different configurations.

5. In your groups of about three, you will now create a ***visual servoing-like*** application to use OpenCV and SSD (sum of squared differences) to move the real Baxter robot to track a moving object in Cartesian space. Hmmm. This is tricky in the arbitrary case, but we will simplify it by assuming the object moves on an x-y plane (such as the ground) and the camera is pointing straight down along the z-axis. (Explain in your lab write-up why this makes it easy.) To do this, you will need to choose an object with a small fiducial monochrome sub-image that is highly distinct. (e.g. a book with a simple image or unique letter in the title.)

   First, command the robot to move into a configuration with the camera on the wrist to point straight down (along –z) at a constant height. You might want to do this in joint space, rather than Cartesian space. (Explain, in your lab write-up, why we might want to do this in joint space. Also, explain if you chose to use joint space or Cartesian space.) You will also want to align the camera so the camera frame x-axis is conveniently aligned with the robot's x-axis.

   Calibrate your camera with a known object (could be a checkboard or lines or something) to generate conversion factors from pixels to millimeters in both axes of the camera frame. (Write up your calibration approach and provide the conversion factors and computations required to find them. Explain why we might need to calibrate both axes.) Set up frames of reference that you can transform between so you can 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. Use your object with a small distinct pattern that is not easily confused (a "fiducial" or "feature" similar to the "good" features examined in the HW and in lecture). Then, move the robot continuously in X and Y, using a callback function, 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 target and move it with key strokes, maybe?) Also, you are recommended to use OpenCV (import “CameraController” and “cv2” and “cv_bridge”) so check out the hints on the webpage. OpenCV 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 – see the wiki.)

For each new image, convert to OpenCV and 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 of visual servoing by tracking in a plane parallel to object motion and allowing clean backgrounds with clear image features with slow motions. This is not true visual servoing because we are not closing the loop in the image plane, but this is a detail for this course. 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 1, problem 5. 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. Include answers to all items in red, above, that are not demonstrations.