## Snap Assembly Position Controlled Demo Instructions

*Note: These instructions have significant changes compared to the previous version. Please re-read carefully.*

1. **Set up Baxter and Workstation**<http://sdk.rethinkrobotics.com/wiki/Baxter_Setup>   
   <http://sdk.rethinkrobotics.com/wiki/Workstation_Setup>   
     
   Note: I have not yet tested version 1.2.0 of the software. When you use the wstool command from above, all your repositories will be set to the latest code. You will need to open the .rosinstall file provide and change the version to: release-1.1.1, except for the simulator. Run catkin\_make again.  
     
   Note: the latest code for Baxter is at version 1.2.0, but I have not had time to test under the new software version. So, you have to go back a few commits
2. **Install our code**Install the birl\_baxter repo by following the wiki: <https://github.com/birlrobotics/birl_baxter/wiki>   
     
   **Note**: you might still get some compilation errors when you run catkin\_make. Try to see if you are not missing any dependencies. If you see the problem is coming from a package inside our demo folder (except for pa\_demo or end\_effector\_teleoperation\_control), ignore that package by creating a file called CATKIN\_IGNORE
3. **Untuck arms**
   1. Run the baxter script:   
      >> cd ~/ros/indigo/baxter\_ws  
      >>./baxter.sh
   2. Run the untuck node  
      >> rosrun baxter\_tools tuck\_arms.py -u
4. **Baxter Base**
   1. Make sure the Baxter base is centered. Use *level marker* on the back of robot to see if balanced or not.
   2. Make sure you have at least 1.5m radius of open air behind baxter and on the sides for calibration routines.
5. **Calibrate the arms**
   1. Remove Grippers if they are on.   
      1. Disconnect the electric connection between the gripper and the arm.  
      2. Carefully unscrew gripper from wrist.
   2. Perform both tare and calibration
      1. Can be done from the screen menu by click on the menu button and selecting calibration and tare, or
      2. Can be done from baxter\_tools:  
         rosrun baxter\_tools calibrate\_arm.py -l right  
         rosrun baxter\_tools calibrate\_arm.py -l left  
         rosrun baxter\_tools tare.py -l right  
         rosrun baxter\_tools tare.py -l left
   3. Placing Grippers Back On  
      For the right hand, when you place the gripper again, make sure the gripper allows the front part of the male camera mold to face in the right direction.
6. **Set up Experimental Table**
   1. Make sure it is as parallel to the floor as possible and very stable – screw/unscrew the legs to extend or contract.
   2. Make sure that the top surface of the table (it can be moved) is aligned with the lower base.   
      **Note**: this table can fall over very easily so be careful.
7. **Place the camera molds**
   1. Female part:   
      There are 2 black dots marked on the experimental side on each side. Place the plastic camera mold there along with 4 screws in the holes to fix the part and make it rigid. Make sure that the female’s part front side is in fact facing to the front.
   2. **Male part:**Use your hand to place the male part inside the male gripper. Make sure to hold the camera part rigidly until it has been fully gripped.
8. Running the code
   1. I have updated the code extensively to facilitate the run of the demo as much as possible.
      1. Make sure you do a git clone origin master inside birl\_baxter.
      2. Then run catkin\_make from the top of your workspace (~/ros/indigo/baxter\_ws)
   2. A new package: pa\_demo has been created.
      1. This package is located under birl\_baxter/birl\_demos/pa\_demo
      2. You can easily go there with roscd, as in: roscd pa\_demo. You can also use roslaunch, rosrun, rosed, etc.
      3. All the files that I previously had placed in baxter\_examples have been moved here. That is:
         1. pa\_openHand.py: opens hands
         2. pa\_closeHand.py closes hands
         3. pa\_jtc\_jointAngles: oldest version. Works on recorded joint angles, no launch file for this one.
         4. pa\_jtc.py: works with KDL, no launch file for this one.
         5. pa\_jtc\_trac\_ik.py: works with KDL or Trac\_IK (though some bugs in there), has a launch file.  
              
            They are all found under the scripts folder.   
              
            You can also find all the documentation under the Documentation folder, you can find parameter files under config, you can find launch files under launch, you can record/find rosbags under bags, and you can find all the nodes under: /scripts/pa\_demo  
              
            pa\_demo  
             ----------bags  
             ----------config  
             ----------documentation  
             ----------include  
             ----------launch  
             ----------scripts  
             ----------src
   3. The code will consists of two parts:
      1. Keyboard teleoperation and a pa\_jtc\_rightArm.py.
         1. Keyboard Teleoperation:  
            - Will be used before running pa\_jtc\_rightArm.  
            - Move to the reference position (tilted contact between the parts) and then manually recorded the endPose and the joint angles. After knowing these values, copy and paste them correctly into pa\_jtc\_trac\_IK.py.  
              
            1. Running the teleoperation launch file:  
            >> roslaunch end\_effector\_teleoperation\_control end\_effector\_marker\_control.launch keyboard:=true   
              
            - When the node starts it will give you a keyboard map, copy it and paste it, into gedit for reference. Try to make contact at an angle between 20-50 degrees.   
              
            2. Getting end pose info:  
            >> rostopic echo -n 1 /robot/limb/right/endpoint\_state/pose  
              
            3. Getting joint angle info (the order is standard: s0,s1,e0,e1,w0,w1,w2)  
            >> rostopic echo -n 1 robot/limb/right/gravity\_compensation\_torques/actual\_position
         2. Run the Pivot Approach State Machine
            1. You will be running a launch file that will do the following:

Run the gripper action server

Run the joint action server

Run the pa\_jtc\_trac\_IK.py with correct topic remappings.   
You can run the launch file as follows:  
>> roslaunch pa\_demo pa.launch

Also, you will be able to pass two arguments: record/debug

record  
- Starts a rosbag record for the important topics. When the terminal receives Ctrl+C, the bag is saved to: birl\_baxter/birl\_demos/pa\_demo/bags  
- The command line for recording is:  
>> roslaunch pa\_demo pa.launch record:=true

debug  
- opens an xterm with pdb on it.   
>> roslaunch pa\_demo pa.launch debug:=true  
But this is not the easiest way to debug. An easier way for you might be to put an ipdb.set\_trace() command before the line that you want to debug. Then run the pa.launch file normally, and the terminal will stop at that line.

**What is the code doing:**

0. Assumes you have collected the reference pose and joint angles and have copied them into pa\_jtc\_track\_IK.py

1. Moves to a starting position conveniently placed directly above of where the female camera model should be  
2. Opens the gripper so you place the camera mold on it  
3. Waits for you to press any key to close the gripper  
4. It computes the following 3 points: goal position, approach1, and approach2.  
5. It moves to start first, then to approach1, 5cm above the reference point, then to approach2, 1 cm above the reference point, then the reference point, then the goal.

6. After it reaches the goal it opens the gripper, and returns to the staring pose.

**Other things to note:**  
1. There are several design parameters at the top of the file that let you test different behaviors. I have selected the parameters that are most stable. But you can experiment with them.   
2. I have included a short summary of testing and evaluation of design parameters in the documentation folder

3. The goal values are also hard coded, I try to find the goal position and keep the ROLL and the PITCH angles stored there. You will have to do this when you set up your environment.

4. You will need to run the experiment many times fine tuning the reference angles. If one set of angles does not work, try them again once or twice more, sometimes Baxter will move better the next time.

**Caution:**  
Keep the e-button close to you at all times. If you feel anything might go wrong press it, then do a ctrl\_c on the terminal. You can then re-activate the arm, move it with zero-g mode, or with one of our recorded\_motions.  
  
**Recorded Motions**   
They can be found at: birl\_recorded\_motions/scripts   
example\_goHome: home position  
example\_paHome\_rightArm.py used for the pa\_demo  
example\_shutdown used to move the arms to a safe place to turn off the robot.

**Turning off the robot**  
- From the menu, or through ssh  
>> ssh [ruser@011405P0002.local](mailto:ruser@011405P0002.local)  
>>passwd: rethink  
>>sudo shutdown -h now  
>> exit