SYSU Advanced Robotics Lab
Baxter Humanoid Robot Demos

Contents

Overview

This documents presents a number of exciting projects to work on as part of your final project in the ROS class.

We would really like all students to perform their final project on the real Baxter!! We will give you as much encouragement as possible to do that.

If you can finish the simulation successfully, then you can try on the real robot. Your project counts for 30% of your final grade (HWs count 70%). If you can get your code to work on the real robot, I will give you 20% extra points on your final grade!! So if your final grade was going to be 80, you would get 100%!!

Regardless of whether you do the simulation only or work on the real robot, we will have a **Project Presentation Day** in which all students will demo their work in A406 together on Saturday, July 4th.

VPN

Many of the demos listed below have links to YouTube. Since most of you do not have access to a VPN directly, please know that a constant VPN connection is available in the Advanced Robotics Lab in room A406 in the school of Software.

You can use the lab space if either a TA or Dr. Juan is present, or alone with special permission from Dr. Juan.

Access to A406 and Teams

During this project you will need extensive access to A406. Each team will be composed of two members. Each team will also have a team leader. The team leader will need to provide Dr. Juan his ID and go to the card office in the School of Software to acquire temporary access to room A406 for the remainder of the semester.

Robot Lab Rules

1. Keep organized and in good condition all the time. Put away tables, chair, and equipment in its proper place.
2. Make sure the door is securely locked after you leave the room.
3. Turn off the light if you are the last one in.
4. If you bring any trash, make sure you take it out!

Robot Accidents

- Please treat all robots with extreme care. Take many precautions.
- Email me before you use any robots any day so I know you are using them.
- If you are not sure if there is a problem with a robot or if something broke contact me immediately (150-1313-7354).

Schedule

Please not the following dates carefully:

1. Open github.com account and send to Dr. Juan by email
2. Project Selection: June 2nd
3. *Recommended Date for finishing Alpha Stage: June 17th.*
4. Project Presentation Day: July 4th.

Deliverables

At the end of the project you are required to

A) Document your demo on the official sysu_baxter github page: and provide a clear English and Chinese explanation of the demo, video, and working code. See the face follower demo as an example:

Dr. Juan will give your read/write permissions on GIT after you provide him with your username. This will allow you push directly to the sysu_baxter repo. Each project is in charge of creating their own package and placing it under the sysu_baxter/demo folder.

B) Present your project during project day.

GitHub

The first step is for you to register on **github.com.**
This is important because I will need to add your username as part of a Team in the sysurobotics github page. So, please sign up and then send this information to my (Dr. Juan's email): , stating your name, email, and github user name. After that, I will add you to the appropriate github team, from where you will be able to see the rest of the class members and each other's repositories.

How to Push the Code to the SYSU Baxter repository

Before we push any code, we need to make sure that we do the following steps:

1. Create or clone your package inside sysu_baxter/demos.

2. If you create the package, use a name as designated below. If you clone the package use the default name.

3. When you clone a package, git will have a repository called origin. That repository belongs to the original person who created that package. We want to **push** our code to a different repository. That is the sysu_baxter repository. To do that we need to follow a number of steps:

a. Move to your package directory, i.e. cd ../sysu_baxter/demos/my_package.

b. Set the URL of the sysu_baxter repository and name it upstream by doing as follows:
$ git remote add upstream

c. To check you have done this correctly, list all your repositories inside sysu_baxter/demos by typing:
$ git remote -v

You should see 4 lines: 2 lines corresponding to the URL for the origin repo (fetch/push), and 2 lines corresponding to the URL for the upstream repo (fetch/push). For example,

```
origin    https://github.com/haxelion/baxter_pickandlearn.git (fetch)
origin    https://github.com/haxelion/baxter_pickandlearn.git (push)
upstream        https://github.com/sysurobotics/sysu_baxter.git (fetch)
upstream        https://github.com/sysurobotics/sysu_baxter.git (push)
```

d. Now that you have added the sysu_baxter repo as upstream we need to push our first file. If you are not very familiar with GIT this is the time to go through their tutorials now. Knowing GIT is incredibly important and powerful and relatively easy to use.

First create a descriptive Readme file of what your demo is about and if your demo was created by someone else make sure to credit them by including their name and github.com IRL. When you have finished, let's select this file to be pushed. First we need to add it. Make sure you are in the local directory where you have the ReadME file.

```
$ git add ReadMe
```

When a file has been added, we say it has been staged.

The second step is to commit this file. When you commit, you are basically saving that file to your local repository. Every time you commit you will create a history point and get a commit number back. Also, each commit needs to be accompanied by a descriptive message. The message is always necessary:

```
$git commit -m “Readme was created. Documents the purpose of this package along with its dependencies and author information.”
```

Finally, after you have committed the file, we need to push it from our local git directory to the online repository. Here we need to push to the sysu_baxter repository not the original one. So when you push you need to establish two things: (I) the repo, in our case *upstream*, and (ii) the branch, in our case the default *master* branch.

```
$ git push upstream master
```

Anytime you make a meaningful change, it's a good time to commit. Before a commit, one can add or stage more than one file. It's important to make sure that we always add only changes that are related to each other. For example, if you changed kinematics files and also joint_trajectory files, don't mix these two together, Add and commit changes in kinematics first, and then add and commit changes in joint_trajectory files second. This is important, because if you ever need to go back in history, you can isolate the changes to a single area. It becomes more difficult when you mix changes in many places.

Documentation

1. Format

Create a tutorial guide in English and in Chinese. The format of the tutorial should follow that of the Rethink Robotics' SDK (see example here:).

Again, an example of such documentation can be found at:

It's very important that you include all the following sections:

- Summary
- Quickstart
- Overview
- Structure
- Video
- FAQ

Markdown Language
These demos are written using a language called markdown language. Markdown is very easy to use and makes your documentation look great. Github supports markdown language natively.

is a webpage where you can learn to use markdown and see how your document looks in real-time. You can test your documentaion here first, or you can also try github's preview function.

Updating the Wiki
When you finish, documenting you can go to: **face_follower_en** and **face_follower_cn**. You can paste and edit the documentation you created in stackedit here.

2. Create a link to your page on the demos page.
Please edit the demos page both in English and Chinese. You can go to each of these two links:

And then insert the appropriate link to your new page. i.e. for the "Face Tracking Demo" it is:
0. [Face Tracking Demo](

2. Video

Your demo must include an (embedded) professional looking video. That means, your video must have:

- Slide 1: A title page at the beginning of the video:
  - name of project,
  - names of authors,
  - name of Professor: Dr. Juan Rojas,

    - name of course: 2015S-ROS, and
    - link to our robotics lab web page:
- Slide 2: A brief description of what the demo is about, and

- Slide 3: A video of the demo with a vocal explanation in English.

The video should be of moderate resolution. Such videos can be made with Windows Movie Maker, Adobe Premier Pro, and others.

Videos should be hosted on a site like youku/youtube (so you can embed them in the wiki page) and also provided them to Dr. Juan on a USB/CD. The name of the video should contain the name of the demo in the file.

Project Choices

There will be one project per two students.

A number of projects will be described below. Some of these projects needed to be created from scratch, others have been created by others before, and need to be implemented by following their instructions (sometimes there may be no instructions). Each project has its own set of challenges, and there may be no single one that is easier than the others.

Students will also be able to suggest their own projects if it meets minimum requirements in interest and complexity as deemed by Dr. Juan.

A list of projects is shown below, with a detailed explanation of them following after that.

Undergraduate Students

1. Integrating Kinect and Baxter
2. Kinect Based Arm Tracking
3. Greet Demo: Baxter Face, Speech Recognition, Face Tracking, and Greet motion.
4. End-Effector Teleoperation Control.
5. Visual Servoing Demo.
6. The Illusion of a Thinking Machine (Not in Simulation)
7. Baxter Pick and Learn

Graduate Students

8. Hybrid Position and Force Controller
9. Collaborative Manipulation
10. Deep Learning and Grasping
11. MoveIt Pick and Place

Project Details

This first demo is not part of your project list, but I include the description here as it may help you in performing some of the other projects.

0. Face Recognition and Head Tracking

This demo can be performed using rbx1 and baxter nodes. The nodes that you will use is the:

• Face Detection: rbx1_vision:		face_tracker2.py
• Tracking the Face:	rbx1_apps:	object_tracker.launch
• Moving Head:	baxter_examples:	head_motion.launch

With rbx1_vision you already have a solution to detect faces. This will return the position of the region of interest that can in turn be read by the object_tracker code. In this case, the head camera should move so as to keep the human always in the center of the camera. So, after we detect a difference, we move the head so as to make that difference zero. This will use basically a P or PD controller. By including a D term, the motion will be smoother.

1. Integrating Kinect and Baxter and testing on MoveIT.

This demo consists of a number of parts:

I. Integrate the Kinect to Baxter
II. Calibrate the Kinect and Baxter,
III. Test the system by using MoveIt and move the arm of the robot while avoiding collision with an observed obstacle.

All relevant information for steps (I) and (III) are found at:

All relevant information for the calibration of the sensor is found at:
. This package has a dependency on AR tags. You can run AR tags easily by learning from Ch.10 in the RBX Volume 2 book (if you have not purchased this book as Dr. Juan for a copy of this chapter).

Where to mount the Kinect?
The Kinect will be mounted on the head of Baxter. It is already placed there. You will need to ensure that the sensor is properly positioned and fixed. To fix the head we can use a double sided tape that does not damage the robot but also guarantees that the sensor will be steady. From that position, you will need to compute the static transform between the sonar frame (the last from on the head) and the Kinect.

Test Case
The test case will be to put a large obstacle on a table in front of Baxter, and then use both the Kinect and Moveit to identify the object and avoid as the arm moves from one side to another.

2. Kinect Based Arm Tracking

This demo can be performed using rbx1 and baxter nodes. Suggested nodes are:

• Skeleton Tracker:	rbx1_vision:	openni_tracker.launch
• Arm command:	baxter_examples:	joint_trajectory_client.py
	baxter_interface	joint_trajectory_action_server.py

- Your interface package/node: baxter_arm_tracking

Coordinating Robot and Human Joints
In this demo, consider identifying the {0,0,0,0,0,0,0} angle position for both the right and left arms of Baxter. Consider having the human imitate approximately the same position and read the angles returned by the skeleton tracker at that position. These values will then need to be used to offset the positions of the human arms.

Your node would map the joint angles from skeleton tracker to the baxter action_client. In this way, when the human arms move, the updated angle positions will be sent to the baxter joint controller and imitate the motion. I believe the skeleton tracker is 6 DoF and Baxter is 7DoF. Double check. But to fix this issue you might fix one joint angle in Baxter.

Speed Response of Arm Tracking
You need to evaluate how fast the joint controller will respond to human motions. If the response is not fast enough, you may consider including not just the joint angle information, but also velocity information. The velocity controller would require for us to compute the velocities of the human arm. That is, for each time step in the computer, find the difference between two consecutive joint angles and divide by the computer_time step. This has to be done for each of the DoF in each arm.

3. Greet Demo: Baxter Face, Speech Recognition, Face Tracking, and Greet motion.

This demo can be performed using rbx1, baxter_examples, and sysu_baxter/demos nodes. The nodes that you will use is the:

• Face Expressions:	baxter_examples	xdisplay_image.py
• Face Tracking	demos	face_follower
• Speech Recognition	rbx1_speech	voice_nav_commands.launch
• Greet Motion	baxter_examples	joint_recorder.py -f greet.dat
		joint_trajectory_action_server.py
		joint_trajectory_file_playback.py –f greet.dat

In this demo, we want Baxter to act socially with guests. There are 4 things we want to accomplish:

- **Baxter's Face** (the display screen on his head)
  We want the face to show different kinds of faces: happy, serious, etc. You can find Baxter faces online and photoshop them to suit your needs, or you can create your own set of professionally looking faces to display on the screen.
- **Face Tracking**
  We want the head always to face the human. You can use existing code in sysu_baxter to accomplish this.
- **Speech Recognition**
  See RBX 1, Chapter 9 to see how to implement Speech Recognition on Baxter. There are two things we want to do here:
  I. **Verbal Communication**
  We want Baxter to hear basic interactions from a human and respond back to him. You should implement about 5 different interactions. They could be: "Hello Baxter, how are you?"; "How old are you?", "Where are you from?", "Good-bye", "What's your favorite sport?", "What's the most difficult thing for you to do?" For each of them you should consider what emotion and face Baxter would display on the screen.

  II. **Body Communication**
  The second part to communication is body expression. We want the arms, grippers, and perhaps head to move in such a way that it makes its reactions very intuitive. So, for example, for a greeting interaction, Baxter could wave his hand or stretch it out for a hand shake.

Arm Motions
Each arm motion can be saved into a file. For example, currently there is a simple greet motion in the sysu_baxter package located in the sysu_baxter/recorded_motions/motion_files. Each motion can be generated using the motion using the joint_recorder.py node and later played using the joint_trajectory_action_server.py and the joint_trajectory_file_playback.py.

4. End-Effector Teleoperation Control.

This demo will teleoperate the end-effector using three different modes: (I) the keyboard, (ii) a PS3 joystick, and (iii) RVIZ Interactive Markers.

There are already existing nodes that do teleoperation using the keyboard and the joystick. Resources on Edmodo will also be provided to learn how to use the interactive markers.

- Joystick: joy joy_node
- Baxter Arm: baxter_examples joint_position_joystick.launch joystick:=ps3
- Inverse Kinematics: example_baxter_kinematics display_FkinIKin.py
- Interactive Markers: for general information see:
  http://wiki.ros.org/interactive_markers

For Rviz tutorials (starting on item3) see:
http://wiki.ros.org/rviz/Tutorials

This demo will use both the keyboard and the ps3 joystick controller found in the A406 lab. The goal is to easily teleoperate the pose of the robot's end-effector using either a keyboard or a joystick.

You can see how the keyboard or joystick are used for teleoperation of the arms using joint angles under the:
baxter_examples/scripts/joint_position_keyboard.py and
baxter_examples/scripts/joint_position_joystick.py.

The difference between this demo and the joint_position_keypord.py and joint_position_joystick.launch demo is that we will control the end-effector. The previous demo controls each individual joint of the arm.

In this project, instead of mapping incremental changes between the joystick/keyboard to the joint angles of the robot, we need to do it to the {x,y,z,R,P,Y} variables. We may want to move the wrist of the robot in the x-direction alone, so we would press a key and the wrist would move in that direction in a straight line. Hence, in this project, you will need to compute a new Cartesian pose (position/orientation) and then pass the pose to the inverse_kinematics call.

Instead of using the IK service in baxter_examples, it will be easier and more robust to use the Inverse Kinematics call used in:
sysu_baxter/examples/example_baxter_kinematics/scripts/display_FkinIKin.py.

This call will convert the pose into appropriate joint angles and move the arms accordingly.

5. Visual Servoing Demo.

   This demo will be an implementation of an already existing example from WPI University. The project consists in Baxter finding balls in one region of the table, grabbing them, and then placing them on an egg tray.

   Students will benefit if they have some basic experience in image processing or want to learn some image processing. All details including the code can be found at:
   http://sdk.rethinkrobotics.com/wiki/Worked_Example_Visual_Servoing.

6. The Illusion of a Thinking Machine (Not in simulation)

   This demo is all about mixing Baxter with a kind of artistic performance and is suitable for those who like acting. The following video in YouTube shows you a great demonstration:
   https://www.youtube.com/watch?v=lfG6_LbAOa4

This demonstration lasts 6 minutes. Yours should last 1 minute. The robot motion should follow a story, include an animated face (publish short gifs to xdisplay) , and speak (use rbx1_speech in RBX1.Ch09).

This demonstration lasts 6 minutes. Yours should last 1 minute. The robot motion should follow a story, include an animated face (publish short gifs to xdisplay) , and speak (use rbx1_speech in RBX1.Ch09).

7. Baxter Pick and Learn

This demo involves sorting pieces spread out in a table onto a special place on the table. All relevant information can be found at the following link:
http://sdk.rethinkrobotics.com/wiki/Baxter_Pick_And_Learn

In addition, to a report on this project that can be found on Edmodo: 2015S-ROS/2015S-ROS-Projects/Demo07 Pick And Learn.pdf

8. Hybrid Position and Force Control (Graduate Student – No work in Simulation)

The goal of this project is to implement a *hybrid position-force controller*. References are available for the implementation. Provided separately.

The goal of this demo is for Baxter to hold a cube (available at the lab) on the grippers of its right hand, then bring the cube down from some position in the environment down to the surface of the table until the cube makes contact with the table such that the cube is parallel to the surface of the table and pushes slightly downwards (~1N in the –z-direction with respect to the world). The next step would be to slide the cube from left to right (y-direction wrt to the world) and from the back of the table to the front of the table (x-direction wrt to the world).

The hybrid position and force controller requires that we use two feedback loop. One for the position and one for the force. The desired position coordinates will be provided in Cartesian Space and the desired forces and moments through a wrench at the wrist.

Baxter already has a built in position controller that we can use along with Inverse Kinematics. But as far as the force controller, we will need to create a controller that converts wrenches to joint torques through a Jacobian matrix, and then use Baxter's built in Effort Controller to move the robot.

Baxter provides the current wrench (forces and torques) at the wrist for either arm through the topic:
`/robot/limb/<side>/endpoint_state`

9. Collaborative Manipulation

This demo lets Baxter hold a ball with two arms while these arms handle the ball in different angles. Both arms always maintain the same relative pose, so the ball never falls.

All code relevant to this demo can be found at:
http://sdk.rethinkrobotics.com/wiki/Collaborative_Manipulation

10. Deep Learning for Detecting Robotic Grasps

This project from Cornell University demonstrates a number of intelligent grasps using Baxter as implemented through a deep learning algorithm. Their work is called: deep grasping.

Site:	http://pr.cs.cornell.edu/deepgrasping/
Video:	https://www.youtube.com/watch?v=f9CuzqI1SkE
Data:	http://pr.cs.cornell.edu/grasping/rect_data/data.php
Code:	http://pr.cs.cornell.edu/deepgrasping/code/deepGraspingCode.zip

11. MoveIt PickandPlace

This demo integrates the use of MoveIt along with the Kinect Sensor and a Pick and Place Library to perform both motion planing and pick and place planning with Baxter for simple blocks on a table.

Dave Coleman has tried this before, but his work is a bit outdated. Try to get this running on the real robot. Refer to this work here: https://github.com/davetcoleman/baxter_cpp/blob/indigo-devel/README.md

You can also see how this looks in the PR2 by watching the following video:
https://www.youtube.com/watch?v=Jk_s98U5ob8

General MoveIt Tutorials are found at:
http://moveit.ros.org/documentation/tutorials/

Pick and Place Tutorial is found at:
http://moveit.ros.org/wiki/PickAndPlace

Schedule

Please type in your schedule for the next 5 weeks:

https://onedrive.live.com/redir?page=view&resid=B203D6C7911C79D0!4455&authkey=!APDWa8SinAWkMUU

Project Day

Project Day will be an opportunity for you to show what you accomplished. This event will be open to the public. All your classmates will be there as well and will look at each other's work. Every team is required to present their project, whether it was in simulation, real-robot, or both.

Date: Saturday, July 4th.
Location: Advanced Robotics Lab A406, School of Software.
Time: 2pm – 6pm.

Each time will have 20 minutes to setup and run their demos.

Baxter Loan Agreement Form

Please Print

First Name (Chinese, Pinyin)	**Last Name (Chinese, Pinyin)**	**Student ID#**
Phone #	**Email**	**2015S-ROS or 2015-RVC**
Equipment Name *Baxter Research Robot + Electric Grippers*	**Distributor Name** *Gaitech Robotics*	**Date** June-July 2015

By accepting this equipment and signing this form, the student acknowledges that the equipment is the property of Sun-Yat Sen University (中山大学）and is loaned to the student as an academic tool. All equipment issued will be used for educational purposes and will be supported by Sun Yat Sen University, School of Software (中山大学软件学院) in that endeavor. However, by signing this agreement, the student accepts responsibility to use reasonable care in the use of this equipment and accepts liability for any actions that may be deemed inappropriate or damaging to the college.

Additionally, it is the student’s responsibility to ensure that all equipment is returned to Sun Yat Sen University, School of Software (中山大学软件学院) at the end of the lab session. If for any reason the equipment is not returned, the student is responsible for the full cost of the broken equipment:

- Baxter Robot: 190,000RMB
- Electric Gripper (each): 12,500RMB.

By signing this agreement, the student agrees to the terms set forth.

Student’s Signature: ______________________Issuer’s Signature:___________________________

Printed name:____________________________ Date: ____________________________