Robotic Software Package for HRI ASD Study
Max Uhlenhuth
Advisors: David Feil-Seifer Ph.D., Brian Scassellati Ph.D.

Abstract
For my senior project, I created a robotic imitation package to be used in a human-robot interaction study. The package takes coordinate data from a Microsoft Kinect sensor, determines the angles between joints, and sends the joint angle to another package. It turned out that my initial goals were a bit on the ambitious side and I did not complete the feedback portion of the package. However, I did implement a joint angle conversion utility as well as a visualization tool. This project was my first experience working with the Robot Operating System (ROS) and it was a great introduction to an important paradigm in computing.

This package will be used in a study to determine whether robots can be used in a therapeutically beneficial way with children who have Autism Spectrum Disorders (ASD). Since prior studies have shown that children with ASD tend to have an affinity for robots, the hypothesis is that a combination robot and “graded queuing” approach may be able to teach these children how to interact in a socially acceptable manner.

I learned a variety of new technologies over the course of this project. Of course, there was a heavy ROS component and I spent a significant amount of my time just figuring out how that ecosystem worked. The primary ROS components I used were the base roscore library, the tf coordinate frame transformation library, the rviz visualization suite, and the rosbag playback/debugging library. I fell victim to a countless number of beginner’s mistakes with the ROS library (notably coordinate transform references and timing issues) but having reached the end of this project, I feel confident that I will be able to tackle further challenges in robotics.

Table of Contents
Abstract
HRI Study
Code Architecture
  angle_calculator.py
  im_vis.py
Screenshots
Lessons Learned
Sources
HRI Study

This project fits into a larger HRI study about how children with Autism Spectrum Disorders (ASD) interact with robots and whether robots could be used to train the children to have socially acceptable interactions. This type of work was the focus of David Feil-Seifer’s dissertation at USC entitled “Data-Driven Interaction Methods for Socially Assistive Robotics: Validation With Children With Autism Spectrum Disorders.” The basic setup of the project is that a robot plays a “Simon Says” type game with an ASD child and offers “graded cueing” feedback if the child poses incorrectly. Feil-Seifer cites Hartman-Maeir (2009) in noting that graded cueing has been recommended for ASD therapy. Feil-Seifer also states that “The documented affinity of children with ASD to robots could lead to more attention” (p 125) which makes a graded-cueing robotic study an obvious next step.

Feil-Seifer states the criteria for the robot as:

For a robot to be able to play this game in such a way that it is effective for therapeutic interaction, it would need to be able to sense the actions of other participants in order to follow the rules of the game, and to give proper feedback during the game. Particularly, the robot needs to know if the other participants are correctly imitating the robot, and if they imitate when the robot says “Simon-says” and not otherwise (p 127).

My task was to implement the sensing and feedback tasks described. The semester only proved long enough for me to implement the sensing - feedback will be left for another time.
Code Architecture
The package I wrote can be found at http://yale-ros-pkg.googlecode.com/svn/trunk/sandbox/imitation/. The bulk of the code is contained in two ROS nodes: angle_calculator.py and im_vis.py

angle_calculator.py
This node is subscribed to the tf channel. It pulls the joint coordinates (sensed by the Kinect) for the left and right shoulders, elbows, and hands. These coordinates are all referenced from the camera frame. The node then uses linear algebra and trigonometry to normalize the shoulder/shoulder, shoulder/elbow, and elbow/hand vectors and determine the joint angles between them.

Shoulder roll and elbow roll were fairly straightforward applications of the law of cosines. Shoulder pitch was a bit of a hack that assumes that the shoulder are parallel to the ground. Dropping a perpendicular from the elbow position to the plane through the shoulders parallel to the ground and using that point for a quick arctan yielded the shoulder pitch angle. Elbow yaw was the trickiest to achieve. The key was to subtract the shoulder/elbow vector from the elbow/wrist vector to normalize the coordinates relative to the frame of reference. Then yaw was a simple application of arctan.

The output of this node is published as a nao_msgs.msg.JointAnglesWithSpeed message to the joint_angles channel so that it can be picked up by the robot model node and the im_vis node.

im_vis.py
This node is used to display the joint angle values in a simple, human readable format. It subscribes to the joint_angles channel and then publishes Marker objects to the imitation_visualization channel so that they can be displayed in rviz.

There is also a launch file named fake_robo.launch that is used to start up all the relevant nodes.
Screenshots

The following screenshots are rviz output showing:

- the Kinect data (the red, green, and blue coordinate frames) for the shoulders, elbows, hands, and head
- the robot model (in gray) representing a convenient visualization of the joint angles
- and the joint angles on a scale from min to max (in red).
Lessons Learned

This entire project was a big learning experience for me. Having never done anything serious in robotics before, this project was a great introduction to not only building robotic systems, but also how robots fit into a larger research context. Attending weekly lab meetings and weekly check-ins with Dr. Feil-Seifer gave me a good sense for how research in the academic world works and how the work I was doing fit into the larger lab projects. It was also exciting to be around for the huge Expedition grant award - hopefully it marks the beginning of great things for robotics at Yale.

The technologies I learned over the course of this project were:

**roscore**
This is package that powers the basic distributed node system of ROS. Learning the theory and application of this paradigm of computation has given me powerful tools that I plan to use in future hobby robotics projects. I also learned all the basics of installing ROS from scratch, creating new ROS packages, installing dependencies, and importing other people’s code.

**tf**
This package contains coordinate transformation utilities that let programmers abstract out much of the detail of working with different coordinate frames. Unfortunately, it’s super tricky to fully understand and I wasted a whole week struggling with an awful tf bug. Turns out I had flipped my target frame and reference frame and that caused strange behavior indeed. Another trick was that tf is very sensitive to timing issues, so you have to reset the clock every time you run it (that was another fun week).

**rviz**
This is the visualization package for ROS. It enabled me to view the Kinect-sensed skeleton, the robot model, and my visualization for angle ranges.

**rosbag**
This is a ROS utility to play back previously collected data. I used it to play back a Kinect recording so that I wouldn’t have to stand in front of a Kinect each time I wanted to test my code.

Sources