NAO-Kinect Teleoperation

Group Members: Jill Mercer & Brady Rainey

Group Number: 3

Usernames: jmm1037 & blr1008

Date: December 1st, 2016

Problem: Teleoperation of NAO using Kinect

1. **Description of Problem**
   1. Operation of NAO’s limbs using an Xbox Kinect.
2. **Description of Solution**
   1. Using Java for Kinect’s Skeleton class we could pull in the X, Y, and Z positions of specified joints. Using the cross product of two vectors in three dimensional Euclidian space we could produce the degree of angle for the elbow and shoulder joints. This gave us the angle we fed to the NAO via left and right Elbow and Shoulder Roll functions.
   2. For the Shoulder Pitch we feed the slope of the upper arm to an arctangent function and then use the resulting radian directly.
3. **Results**
   1. We came very close to 1:1 mirroring of the NAO via Kinect. Some issues we were unable to overcome are related to NAO’s collision avoidance. If mirroring would result in the NAO coming in contact with itself the NAO’s actions become unpredictable. Sometimes this would result in the NAO stopping short and other times the NAO would continue moving, such as moving its arm above its head. Both would end in the NAO becoming out of sync with the human.
4. **Code Reusability**
   1. **GetVectorAngle**
      1. This function takes a Skeleton Object and three joints. Much of the code involves interpreting the 3 joints into X, Y, and Z numbers. If this was moved to a higher level the function could be generalized to accept these 9 points and return a vector angle and re-used in any code requiring vector angle math.
   2. **KinectTracking**
      1. This code is highly targeted to the NAO robots. The angle calls are based on the NAO API and include calculations to turn the angles into the angles required for the NAO. For example, with the elbow angle the vector angle returned for a straight arm is 180 degrees. The angle to send to the NAO for a straight arm is zero degrees. This requires a “massaging” of the data returned before giving it to the NAO robot. These changes fall into two categories, subtracting a constant, such as 180 in the previous example, and negating the angle altogether. This was required for angles moving to the right, such as the right shoulder roll and left elbow roll, because of the way NAO calculates it’s X/Y/Z positions centering on itself.
5. **Future Plans**
   1. Legs
      1. Mirroring the legs may not be an option for the NAO. The NAO cannot smartly balance itself, but there is an option to avoid falling over. This means any action given to the NAO that would require it to move outside its “support polygon” would be ignored and would cause many the leg actions to be discarded.
   2. Kinect 2.0 (Xbox One Kinect)
      1. The Kinect 2.0 adds support for finger tracking as well as other improvements over the Kinect 1.0. This will allow for the teleoperation of the NAO’s fingers.