**Nao Imitation of Human Upper Body Movements Using the Kinect**

*Team 2 - Peter Brook, Johnson Goh, Ryan Oman, Chieh-Han Wu*

**Project Goals:**

The goal of the project is to get the NAO robot to imitate in near real-time the gestures (motion of both arms) of a human using the data from a Kinect camera. The robot and human demonstrator will both be standing, facing each other. To accomplish this task, we will need to: (1) extract joint angles for the arms from Kinect data, (2) create a mapping from these joint angles to the robot joint angles, and (3) use the mapped joint angles to control the NAO in real time.

**Design:**

There are three main components in the design of our system: the Kinect interface, the Human-NAO skeleton mapper, and the NAO controller. The Kinect interface receives the joint data from the Kinect and translates it into an upper body skeleton. The Human-NAO skeleton mapper takes the human skeleton as input from our Kinect interface and translates joints from the human skeleton onto the NAO skeleton. The NAO controller updates the motors onto the NAO with scaled values from the Human-NAO skeleton mapper.

* **Kinect Interface**
  + **Input**: List of joints and their position relative to the Kinect
  + **Output**: Human Skeleton class, which is a collection of joints and their angles with relation to different degrees of freedom.
  + **Description**: This layer translates the joints give from the Kinect into a NAO friendly representation of a human skeleton, meaning that we took the NAO’s degrees of freedom into account when determining how to extract the human skeleton angles.
  + **Joints**:
    - **Shoulders**: We decided to represent the human shoulder as having three degrees of freedom. The degrees of freedom were chosen so that they would map well to the NAO skeleton.
      * *Shoulder Pitch*: This value changes when our hand moves straight out in front of our body. The angle for this motion is computed by projecting the upper arm vector onto a plane which has the vector that goes from shoulder to shoulder as the normal vector. We then use the vector perpendicular to the shoulder to shoulder vector and the spine vector as a reference for 0 degrees.
      * *Shoulder Roll*: This value changes when our hand moves away from our body in a way that is in the same direction as the vector that goes between our shoulders. This angle for this motion is computed by projecting the upper arm vector onto a plane that is parallel to both the upper arm and the collarbone. We then use the down vector rotated around the collarbone with respect to the shoulder pitch (since the plane moves with pitch) as a reference for 0 degrees.
      * *Shoulder Yaw*: If we start with our shoulder roll and elbow roll at a 90 degree angle, and our hands pointing towards the sky (like you are lifting something above your head), this value changes when we rotate our hands forward. On the human skeleton, the shoulder is rotating in its socket, and no other joints are moving. This angle is computed by projecting the vector from the elbow to the wrist onto a plane that has the normal vector of the upper arm. We then use the up vector rotated based on the shoulder roll and pitch to as a reference for 0.
    - **Elbows**: The human body only has one simple degree of freedom for elbows.
      * *Elbow Roll*: If we start with our arms straight out to the sides like a “T”, this values changes when we bend our elbow to touch our shoulder. This angle is computed by taking the angle between the upper and lower forearms, which is always a positive value.
* **Human-NAO skeleton mapper**
  + **Input:** Human skeleton class (collection of joints and their angles)
  + **Output**: NAO skeleton class (collection of joints and their angles)
  + **Joints**:
    - **Shoulders:**
      * *Human Shoulder Roll -> NAO Shoulder Roll*: To scale this motion, we just needed to limit the human motion to a slightly smaller range than normal, since the robot can only go up to slightly less than 90 degrees on without also changing shoulder pitch.
      * *Human Shoulder Pitch -> NAO Shoulder Pitch*: Similar to shoulder roll, this motion was simple to scale and had a one to one mapping from human to robot.
    - **Elbows:**
      * *Human Elbow Roll -> NAO Elbow Roll*: This motion was scaled according to the limits of the human joint, which were defined with the robot joints in mind. The angle between the forearm and upper arm cannot be less than 90 degrees on the robot, so we similarly defined the limit of the human elbow roll to be the same.
      * *Human Shoulder Yaw -> NAO Elbow Yaw*: This motion was a one to one mapping with scaling from what is the shoulder rotating in its socket to the NAO’s elbow rotating.
* **NAO Controller**
  + **Input:** NAO skeleton class (collections of joints and their angles)
  + **Output:** Valid values for the NAO joint motors
  + **Description:** The controller essentially took the joint angles out of the NAO skeleton class, connected to the NAO, and passed the values through.

**Results/Evaluation:**

Individually, the joints performed well when all other joint motor angles were held constant, but when combined together, our solution seemed to break down. One of the main problem areas was the shoulder roll. When tested by itself, it performed reasonably well at moving in the correct direction; however, when shoulder pitch was enabled, shoulder roll would no longer be detected. We believe this occurred because of problems with the Kinect detecting the joint angles and applying our transformations over them. We were trying to map shoulder roll directly onto the NAO, which is not possible because the human shoulder has a much higher roll than the NAO, and the NAO requires a combination of pitch and roll change to map all human shoulder roll motions. This was also a problem with some of the other joints. Direct mappings from the human skeleton to the NAO skeleton caused the NAO to move to odd angles at some points. One other problem with our approach is that many of the motions were off scale. For example when our arms were at our sides, the NAO’s arms were sticking out slightly. This could be solved by fine tuning the human ranges to better fit the NAO motion range.

One aspect that could be extended is smoothing for the NAO movements. Many of the NAO movements were rather jerky, since it automatically updates to whatever the latest human data from the Kinect is. This can be a problem because the Kinect data can be rather noisy, meaning that despite the human movement being a smooth gesture, the Kinect will read a rather stuttered on, causing the NAO to perform a similarly jerky movement. To solve this, we could introduce a delay on when the NAO receives Kinect data, so that smoothing could be applied to the Kinect joint angle values.

Another aspect that we spent a bit of time on, but did not end up implementing, is using machine learning to map the human skeleton to the NAO skeleton. This could be done by first collecting gesture data from a human in front of the Kinect, and then moving the robot arms manually while recording all changes in angles. This would essentially be the training data for a neural network with the input being the human skeleton joint values and the output being the NAO skeleton joint values. This would allow the NAO to learn how to move based off of training data it received from human movements and the data collected through manual movement of the arms.

**Responsibilities:**

Peter - Human Interface, integration, debugging

Johnson - Human-NAO skeleton mapper, report

Ryan - Human-NAO skeleton mapper, integration, debugging, report

Chieh-Han – Kinect Interface, debugging

**Links:**

Code Repository: <https://github.com/omanamos/kinect-nao>

Wiki: <https://github.com/omanamos/kinect-nao/wiki/_pages>