I. Approach

Our original plan was to utilize ROS on Linux to interface with the Kinect, but we were unable to get the drivers to work with our model of Kinect and opted instead to use the Kinect SDK for Windows. After analyzing this technology for some time, we decided that the best approach to the problem was to break it down into three smaller subproblems: raw data acquisition, production of models for use with libSVM, and gesture matching. The first two deal with specifying the gestures that were ultimately used in the system, while the last actually recognized them. Each of these subproblems was dealt with by its own program, which we will detail here.

A. Raw Data Acquisition

The first program gathered joint data from the Kinect and placed it in a file. After initializing the Kinect and setting it up to capture skeleton data, the program entered into a data capture loop. Here, the program waits until it starts receiving meaningful data. Once it does, it outputs the data to a file and to the screen, waits for 200 milliseconds, and repeats the process until 25 frames of data have been captured. The file is formatted with each line containing the frame number, joint number, x coordinate of the joint, y coordinate, and z coordinate.

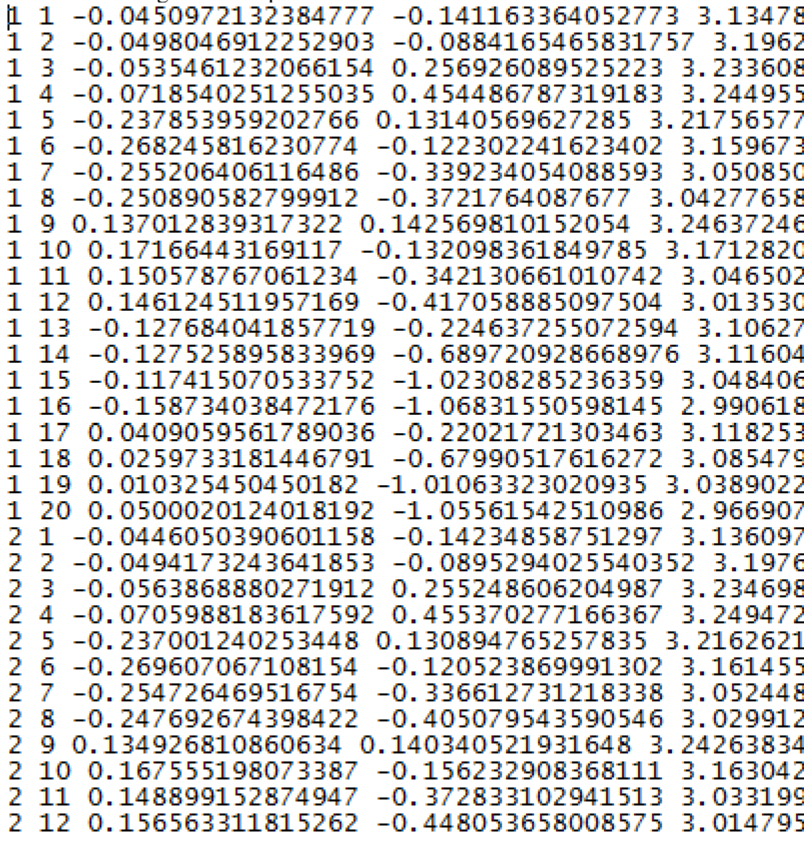
Because all of our gestures are static, 25 frames seemed like a good amount of data to collect. It is small enough that the actor does not have to stand for too long, yet large enough to capture some good data. In retrospect, it may have helped to improve our accuracy if we had experimented with which number of frames gave us the best accuracy.

The Kinect does not always produce good data; on occasion, the skeleton will appear "jittery." To help rectify this, we wait 200 milliseconds between frames. This reduces some of that jitter, which important for our training set in particular. It also allows the program operator to quickly browse over some of the incoming data to ensure that it appears to be meaningful.

B. Production of Models For Use With LIBSVM

C. Gesture Matching

Figure 1. A portion of the raw data file.



```
1 1 -0.0450972132384777 -0.141163364052773 3.13478
1 2 -0.0498046912252903 -0.0884165465831757 3.1962
1 3 -0.053546123206154 0.25692608952523 3.233608
1 4 -0.0718540251255035 0.454486787319183 3.244955
1 5 -0.237853959202766 0.13140569627285 3.21756577
1 6 -0.268245816230774 -0.122302241623402 3.159673
1 7 -0.255206406116486 -0.339234054088593 3.050850
1 8 -0.250890582799912 -0.372176408767 3.04277658
1 9 0.137012839317322 0.142569810152054 3.24637246
1 10 0.17166443169117 -0.132098361849785 3.1712820
1 11 0.150578767061234 -0.342130661010742 3.046502
1 12 0.146124511957169 -0.417058885097504 3.013530
1 13 -0.127684041857719 -0.224637255072594 3.10627
1 14 -0.127525895833969 -0.689720928668976 3.11604
1 15 -0.117415070533752 -1.02308285236359 3.048406
1 16 -0.158734038472176 -1.06831550598145 2.990618
1 17 0.0409059561789036 -0.22021721303463 3.118253
1 18 0.0259733181446791 -0.67990517616272 3.085479
1 19 0.010325450450182 -1.01063323020935 3.0389022
1 20 0.0500020124018192 -1.05561542510986 2.966907
2 1 -0.0446050390601158 -0.14234858751297 3.136097
2 2 -0.0494173243641853 -0.0895294025540352 3.1976
2 3 -0.0563868880271912 0.255248606204987 3.234698
2 4 -0.0705988183617592 0.455370277166367 3.249472
2 5 -0.237001240253448 0.130894765257835 3.2162621
2 6 -0.269607067108154 -0.120523869991302 3.161455
2 7 -0.254726469516754 -0.336612731218338 3.052448
2 8 -0.247692674398422 -0.405079543590546 3.029912
2 9 0.134926810860634 0.140340521931648 3.24263834
2 10 0.167555198073387 -0.156232908368111 3.163042
2 11 0.148899152874947 -0.372833102941513 3.033199
2 12 0.156563311815262 -0.448053658008575 3.014795
```