Kinect Space Invaders

Nate Lane, Michael Harvey, Charles Tandy

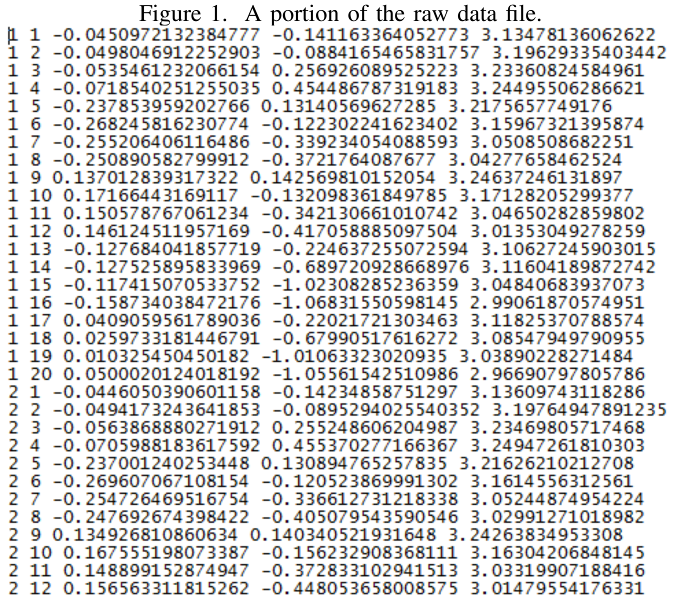
***Abstract*— Abstract stuff here.**

I. Introduction

II. 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.

Figure 1. A portion of the raw data file.



```
1 1 -0.0450972132384777 -0.141163364052773 3.13478136062622
1 2 -0.0498046912252903 -0.0884165465831757 3.19629335403442
1 3 -0.0535461232066154 0.256926089525223 3.23360824584961
1 4 -0.0718540251255035 0.454486787319183 3.24495506286621
1 5 -0.237853959202766 0.13140569627285 3.2175657749176
1 6 -0.268245816230774 -0.122302241623402 3.15967321395874
1 7 -0.255206406116486 -0.339234054088593 3.0508508682251
1 8 -0.250890582799912 -0.3721764087677 3.04277658462524
1 9 0.137012839317322 0.142569810152054 3.24637246131897
1 10 0.17166443169117 -0.132098361849785 3.17128205299377
1 11 0.150578767061234 -0.342130661010742 3.04650282859802
1 12 0.146124511957169 -0.417058885097504 3.01353049278259
1 13 -0.127684041857719 -0.224637255072594 3.10627245903015
1 14 -0.127525895833969 -0.689720928668976 3.11604189872742
1 15 -0.117415070533752 -1.02308285236359 3.04840683937073
1 16 -0.158734038472176 -1.06831550598145 2.99061870574951
1 17 0.0409059561789036 -0.220217213034630 3.11825370788574
1 18 0.0259733181446791 -0.67990517616272 3.08547949790955
1 19 0.010325450450182 -1.01063323020935 3.03890228271484
1 20 0.0500020124018192 -1.05561542510986 2.96690797805786
2 1 -0.0446050390601158 -0.14234858751297 3.13609743118286
2 2 -0.0494173243641853 -0.0895294025540352 3.19764947891235
2 3 -0.0563868880271912 0.255248606204987 3.23469805717468
2 4 -0.0705988183617592 0.45537027716636 7 3.24947261810303
2 5 -0.237001240253448 0.130894765257835 3.2162621021270 8
2 6 -0.269607067108154 -0.120523869991302 3.161455631256 1
2 7 -0.254726469516754 -0.336612731218338 3.05244874954224
2 8 -0.247692674398422 -0.405079543590546 3.02991271018982
2 9 0.134926810860634 0.140340521931648 3.24263834953308
2 10 0.167555198073387 -0.156232908368111 3.16304206848145
2 11 0.148899152874947 -0.372833102941513 3.03319907188416
2 12 0.156563311815262 -0.448053658008575 3.01479554176331
```

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

The second program processed the raw data that was read in from the kinect using a simple star skeleton approach. The processed data was then read into libSVM to produce a model. This model could then be used to predict gestures.

To process the data for libSVM we had to first calculate the angles and distances between the center reference point and each of the five points on the star skeleton. After calculating the distance and angles, we stored the data into histograms made up of a set number of bins.

After the data was processed it was stored in a file. This file was then read directly into LIBSVM to produce the model for our gestures.

C. Gesture Matching

The third program was used to recognize ges-

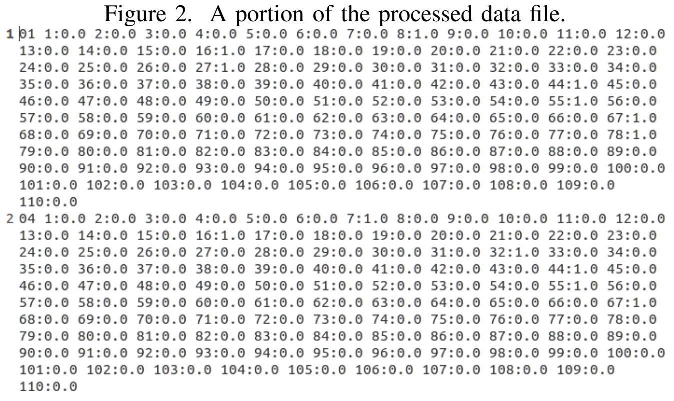


Figure 2. A portion of the processed data file.

```
01 1:0.0 2:0.0 3:0.0 4:0.0 5:0.0 6:0.0 7:0.0 8:1.0 9:0.0 10:0.0 11:0.0 12:0.0
13:0.0 14:0.0 15:0.0 16:1.0 17:0.0 18:0.0 19:0.0 20:0.0 21:0.0 22:0.0 23:0.0
24:0.0 25:0.0 26:0.0 27:1.0 28:0.0 29:0.0 30:0.0 31:0.0 32:0.0 33:0.0 34:0.0
35:0.0 36:0.0 37:0.0 38:0.0 39:0.0 40:0.0 41:0.0 42:0.0 43:0.0 44:1.0 45:0.0
46:0.0 47:0.0 48:0.0 49:0.0 50:0.0 51:0.0 52:0.0 53:0.0 54:0.0 55:1.0 56:0.0
57:0.0 58:0.0 59:0.0 60:0.0 61:0.0 62:0.0 63:0.0 64:0.0 65:0.0 66:0.0 67:1.0
68:0.0 69:0.0 70:0.0 71:0.0 72:0.0 73:0.0 74:0.0 75:0.0 76:0.0 77:0.0 78:1.0
79:0.0 80:0.0 81:0.0 82:0.0 83:0.0 84:0.0 85:0.0 86:0.0 87:0.0 88:0.0 89:0.0
90:0.0 91:0.0 92:0.0 93:0.0 94:0.0 95:0.0 96:0.0 97:0.0 98:0.0 99:0.0 100:0.0
101:0.0 102:0.0 103:0.0 104:0.0 105:0.0 106:0.0 107:0.0 108:0.0 109:0.0
110:0.0
04 1:0.0 2:0.0 3:0.0 4:0.0 5:0.0 6:0.0 7:1.0 8:0.0 9:0.0 10:0.0 11:0.0 12:0.0
13:0.0 14:0.0 15:0.0 16:1.0 17:0.0 18:0.0 19:0.0 20:0.0 21:0.0 22:0.0 23:0.0
24:0.0 25:0.0 26:0.0 27:0.0 28:0.0 29:0.0 30:0.0 31:0.0 32:1.0 33:0.0 34:0.0
35:0.0 36:0.0 37:0.0 38:0.0 39:0.0 40:0.0 41:0.0 42:0.0 43:0.0 44:1.0 45:0.0
46:0.0 47:0.0 48:0.0 49:0.0 50:0.0 51:0.0 52:0.0 53:0.0 54:0.0 55:1.0 56:0.0
57:0.0 58:0.0 59:0.0 60:0.0 61:0.0 62:0.0 63:0.0 64:0.0 65:0.0 66:0.0 67:1.0
68:0.0 69:0.0 70:0.0 71:0.0 72:0.0 73:0.0 74:0.0 75:0.0 76:0.0 77:0.0 78:0.0
79:0.0 80:0.0 81:0.0 82:0.0 83:0.0 84:0.0 85:0.0 86:0.0 87:0.0 88:0.0 89:0.0
90:0.0 91:0.0 92:0.0 93:0.0 94:0.0 95:0.0 96:0.0 97:0.0 98:0.0 99:0.0 100:0.0
101:0.0 102:0.0 103:0.0 104:0.0 105:0.0 106:0.0 107:0.0 108:0.0 109:0.0
110:0.0
```

tures. Upon initialization, the model created from the second program is loaded into libSVM. Then, for each frame of data produced by the Kinect, our program loads each joint into a data structure that specifies its position. This joint is added to a list that contains all joint data for that particular frame. This list is then added to another list, which is a collection of joint data for a series of frames. Once the size of the latter list reaches 25 (i.e., 25 frames have been analyzed), we run some code from program 2 on it to place the data into a histogram, which is then loaded into libSVM for analysis. Each bin of the histogram is numbered and has a value. To allow libSVM to interpret the data, we paired the number of the bin with the value inside it; this pair is referred to as a node. Once a full set of nodes has been obtained, the program uses the model it loaded earlier to make a “prediction” as to what action was performed in the 25 frames being analyzed. The prediction comes in the form of a number corresponding to the gesture being recognized. At this point, we use a switch statement to detect the gesture and send the proper keyboard signals to the window in focus.

III. Experiments

To build a training set, the three of us each performed all seven gestures three times. In our testing set, we all performed each gesture only once. While we believed at the time that this would be sufficient, as we are all of very different builds, it actually proved to be insufficient, as we will explain later.

We performed these gestures while the data acquisition program was running, allowing us to construct primitive data files. We then fed these data files into the second program, the formatter, to produce a model that could be used with libSVM. In our final product, this provided an accuracy of about 80%.

We did encounter some difficulty in acquiring this data. In some of our initial trials, while we were in proper range for the Kinect to read our skeletons in a neutral position, our hands and arms would be out of the frame when raised. Also, sometimes the Kinect would simply not detect our forms, and we would have to redo the experiment.

We produced three testing sets, each of us went through all of the gestures once to produce a testing set. These were the sets that were used to obtain the accuracy of our gesture detection. Our results found the accuracy to be 80.9524%. The results show that the program struggled with telling the difference between gestures 2(right arm out) and 4(right arm up). This is most likely due to our relatively small training set.

In order to get the accuracy of our gesture detection, we obtained our training and testing sets produced by the second program. We then used the easy.py program in libSVM to read these files and obtain the accuracy and the optimal C and gamma values. These values were found to be 8192.0 and 0.03125 respectively. To further increase our accuracy we experimented with different numbers of bins in the histograms from the second program. We found that 11 bins gave us optimal results.

Finally, we ran our program with the game space invaders. We found the gestures with the right hand seamed to work well, but there was difficulty with detecting gestures with the left hand. Again, this was most likely due to our small training set. Given a larger training set we are confident that our program would work much better.

IV. Conclusion

Appendix

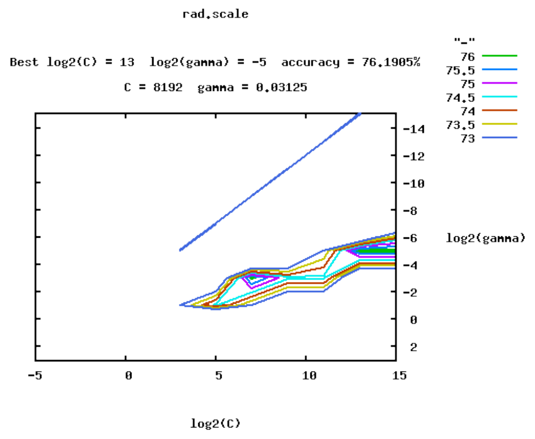
Work Load:

- Nate Lane: todo
- Michael Harvey:
- Charles Tandy:

Figure 3. Confusion Matrix

Expected Result	Actual Result
4	4
1	1
6	6
4	2
3	3
7	7
6	6
5	5
6	6
1	1
2	4
4	2
7	7
3	3
5	5
3	5
2	2
1	1
2	2
7	7
5	5

Figure 4. graph of the grid search


rad.scale
Best log2(C) = 13 log2(gamma) = -5 accuracy = 76.1905%
C = 8192 gamma = 0.03125
"-"
log2(gamma)
log2(C)