Breaking the Status Quo: Improving 3D Gesture Recognition with Spatially Convenient Input Devices

Michael Hoffman*
University of Central Florida

Paul Varcholik†
University of Central Florida

Joseph J. LaViola Jr.‡
University of Central Florida

Abstract

We present a systematic study on the recognition of 3D gestures using spatially convenient input devices. Specifically, we examine the linear acceleration-sensing Nintendo Wii Remote coupled with the angular velocity-sensing Nintendo Wii MotionPlus. For the study, we created a 3D gesture database, collecting data on 25 distinct gestures totalling 8500 gestures samples. Our experiment explores how the number of gestures and the amount of gestures samples used to train two commonly used machine learning algorithms, a linear and AdaBoost classifier, affect overall recognition accuracy. We examined these gesture recognition algorithms with user dependent and user independent training approaches and explored the affect of using the Wii Remote with and without the Wii MotionPlus attachment.

Our results show that in the user dependent case, both the AdaBoost and linear classification algorithms can recognize up to 25 gestures at over 90% accuracy, with 15 training samples per gesture, and up to 20 gestures at over 90% accuracy, with only five training samples per gesture. In particular, all 25 gestures could be recognized at over 99% accuracy with the linear classifier using 15 training samples per gesture, with the Wii Remote coupled with the Wii MotionPlus. In addition, both algorithms can recognize up to nine gestures at over 90% accuracy using a user independent training database with 100 samples per gesture. The Wii MotionPlus attachment played a significant role in improving accuracy in both the user dependent and independent cases.

Index Terms: I.6.3 [Computing Methodologies]: Methodologies and Techniques—Interaction Techniques; I.5.2 [Pattern Recognition]: Design Methodology—Classifier Design and Evaluation; K.8 [Computing Milieux]: Personal Computing—Games

1 Introduction

The use of three-dimensional (3D) gestures is a commonly used approach for interacting in virtual and augmented environments for tasks such as navigation, selection, manipulation, and system control [2]. However, the challenge with using 3D gestures is machine learning algorithms used to recognize them in real time must be highly accurate. In addition, samples of these gestures are needed to train the recognition systems so they are ready for application-level use. The number of training samples needed is often dependent on how many gestures there are in the gesture set as well as whether user dependent or user independent gesture recognition is required.

With the advent of spatially convenient input devices [18], the ability to use 3D gesture recognition in virtual environments is becoming mainstream, especially in the video games domain. Spatially convenient input devices provide

*e-mail: mhoffman.ucf@gmail.com
†e-mail:pvarchol@ist.ucf.edu
‡e-mail:jjl@eecs.ucf.edu

Figure 1: A user providing data for our 3D gesture database.

- 3D input data; be it incomplete partial, error-prone or conditional
- a range of useful sensors, emitters and interface implements
- an inexpensive cheap, durable, easily configurable and robust solution.

Although these devices provide 3D input data that can be used for 3D gesture recognition, they often sense motion, using accelerometers and angular rate gyroscopes, rather than actual position and orientation. Devices that use accelerometers and/or angular rate gyroscopes can suffer from compounding errors due to drift, have no specific frame of reference, and movement rate can produce different acceleration and/or velocity profiles for a given gesture. Given these issues, performing 3D gesture recognition with spatially convenient devices can be a challenging task.

The focus of this work is to better understand how accurately we can recognize 3D gestures using these devices. In particular, we aim to explore how the relationship between the number of gestures, the amount of training samples, and whether recognition is user dependent or independent affects overall recognition accuracy. There have been several attempts made at exploring 3D gesture recognition using accelerometers and gyroscopes (see Section 2). However, to the best of our knowledge, there have been no robust, systematic studies on 3D gesture recognition using spatially convenient devices. Important questions remain regarding these device types and their ability to adequately recognize a collection of 3D gestures. By examining these issues, we hope to ultimately answer the question of how many 3D gestures can these devices recognize with high accuracy and a minimum amount of training. This knowledge will provide guidelines for 3D user interface developers and designers on how to best make use of 3D gesture recognition with spatially convenient devices.

In this paper, we present a systematic study of 3D gesture recognition performance of arguably the most common spatially convenient input device, the linear acceleration-sensing Nintendo Wii

Remote (Wiimote) coupled with the angular velocity-sensing Nintendo Wii MotionPlus. We examine recognition accuracy as a function of the number of gestures in the gesture set and the amount of training samples used to train the recognition algorithm. To perform gesture recognition, we implemented two commonly used machine learning techniques; a linear classifier and a classifier based on the AdaBoost framework. A 3D gesture database was collected for training and testing, consisting of 8,500 gesture samples (see Figure 1). We also examined the effect of using the Wii Remote with and without the Wii MotionPlus attachment and explore recognition accuracy depending on whether a user independent or user dependent recognizer was employed.

In the next section, we examine related work on using linear accelerometers and angular rate gyroscopes in 3D gesture recognition. Section 3 presents a brief description of the two machine learning algorithms we tested and their associated feature set. Section 4 describes the gesture set we used for testing followed by Section 5 which describes our study. Sections 6 and 7 present our study results and a discussion. Finally, Section 8 concludes the paper.

2 Related Work

3D gesture recognition in virtual and augmented environments has been an important research topic over the years. Although there has been a significant amount of work on recognizing 3D gestures using traditional position and orientation tracking devices [3, 10, 16, 17], the use of accelerometer and gyroscope-based devices for 3D gesture recognition has been sparse. Beedkar and Shah experimented with classifying four gestures using a Hidden Markov Model (HMM). The accelerometer data was gathered using numerous TAIYO SPP Bluetooth Accelerometer devices attached to the hands and feet of the participants. They concluded that 25 training samples per gesture were needed in order to achieve 96% accuracy [1]. Kratz also used HMMs to classify a small gesture set [7]. Kratz detailed experiments for game play input using five gestures over a range of 1 to 40 training samples. In their experiments, they utilized accelerometer data from a Wiimote as input for the classifier. The results showed a significant overall increase in average accuracy, 23% to 93%, when using between 1 and 10 training samples. As the number of samples increased from 10 to 40 a less than linear increase was achieved, leaving the authors to conclude that 15 samples at 93% recognition was an ideal configuration. This work tested a relatively small amount of gestures and concluded that at least 15 samples were needed to achieve sufficient accuracy.

Pylvanainen applied HMMs to a slightly larger set of 10 gestures [11]. Both gesture dependent and independent training was conducted with data collected from a hand held mobile device containing accelerometers. The gesture dependent results suggested that only three training samples per gesture were needed to obtain 96.76% accuracy. While the gesture independent tests required a total of 21 (three samples from each of the seven participants) training samples per gesture to reach 99.76% accuracy. Pylvanainen does note that the higher accuracy seen in the user independent test was unexpected but representative of the minimal training samples used in the dependent experiment. Rehm used a Wiimote to recognize 3D gestures based on cultural specific interactions [12]. They conducted both user dependent and user independent experiments on a digit gesture set (10 gestures), a German emblem gesture set (7 gestures), and a VCR control gesture set (8 gestures) using Nearest Neighbor and Näive Bayes classifiers. For the digit gesture set, they claim accuracy as high as 100% for the nearest neighbor classifier in both the user dependent and independent cases and 58% accuracy for the Näive Bayes classifier in the user independent case. Accuracy for the German emblem gesture was 94% and 88% for Nearest Neighbor and Näive Bayes respectively in the user dependent case. For the VCR control gesture set, recognition accuracy was approximately 99% for both the Näive Bayes and Nearest Neighbor classifiers in the user dependent case. Mäntyjärvi also looked at 3D gesture recognition with 8 gestures using HMMs and found recognition accuracy in the high 90s as well [9].

Work has been done that has used similar gestures to the gestures we use in our experiments. Schlomer and colleagues experimented with classifying five gestures performed with the Wiimote using HMMs. The gesture dependent results show that using 10 samples for training and the remaining five for recognition, resulted in classification accuracy between 85 to 95%. For the gestures Square, Circle and Z (also used in our study), the mean accuracy of 88.8%, 86.6%, and 94.3% were shown respectively [15]. Similarly, and most relevant to our work, Kallio's collection of 16 gestures had 6 gestures in common with our set: Line to Right, Line to Left, Line Up, Line Down, Triangle, Parry (although Kallio mentions them by a different name). However, the 16 gestures can be broken up into four distinct gestures in four different orientations. The gestures are composed of time series data obtained from three acceleration sensors placed in a small wireless device. Then, by using HMMs, this work showed accuracy levels nearing 90%, with less than 10 training samples per gesture. However once the number of training samples was increased to 20, accuracy levels rose to over 95%. Kallio also discussed classification confusion between the Line to Right and Parry gestures when providing two samples for training [6]. To the best of our knowledge, this paper presents the first 3D gesture recognition study using the Wiimote and Wii MotionPlus attachment that examines recognition accuracy for up to 25 distinct gestures.

3 Machine Learning Algorithms

To explore 3D gesture recognition performance using the Wiimote and Wii MotionPlus, we chose to examine two different machine learning algorithms, a simple linear classifier and a classifier based on the AdaBoost framework. Although there are a large variety of machine learning algorithms we could have used [4], we chose these two because of their ease of implementation, the fact that they both are feature-based, and that they have been shown to create accurate recognizers using a small amount of training data in other domains [8, 13]. For both algorithms, we adapted them to work with 3D gesture data.

3.1 Linear Classifier

The linear classifier we used is based on Rubine's gesture recognition algorithm [13]. Given a feature vector $\vec{f}$, associated with a given 3D gesture g in a gesture alphabet C, a linear evaluation function is derived over the features. The linear evaluation function is given as

$$g_c = w_{c0} + \sum_{i=1}^{F} w_{ci} f_i \tag{1}$$

where $0 \leq c < C$, F is the number of features, and w_{ci} are the weights for each feature associated with each gesture in C. The classification of the correct gesture g is the c that maximizes g_c.

Training of this classifier is done by finding the weights w_{ci} from the gesture samples. First, a feature vector mean $\bar{f}_c$ is calculated using

$$\bar{f}_{ci} = \frac{1}{E_c} \sum_{e=0}^{E_c-1} f_{cei} \tag{2}$$

where f_{cei} is the i^{th} feature of the e^{th} example of gesture c and $0 \leq e < E_c$ where E_c is the number of training samples for gesture c. The sample covariance matrix for gesture c is

$$\Sigma_{cij} = \sum_{e=0}^{E_c-1} (f_{cei} - \bar{f}_{ci})(f_{cej} - \bar{f}_{cj}). \tag{3}$$

The Σ_{cij} are averaged to create a common covariance matrix

$$\Sigma_{ij} = \frac{\sum_{c=0}^{C-1} \Sigma_{cij}}{-C + \sum_{c=0}^{C-1} E_c}. \quad (4)$$

The inversion of Σij then lets us calculate the appropriate weights for the linear evaluation functions,

$$w_{cj} = \sum_{i=1}^{F} (\Sigma^{-1})_{ij} \bar{f}_{ci} \quad (5)$$

and

$$w_{c0} = -\frac{1}{2} \sum_{i=1}^{F} w_{ci} \bar{f}_{ci} \quad (6)$$

where $1 \leq j \leq F$.

3.2 AdaBoost Classifier

The AdaBoost algorithm we used is based on LaViola's pairwise AdaBoost classifier [8]. AdaBoost [14] takes a series of weak or base classifiers and calls them repeatedly in a series of rounds on training data to generate a sequence of weak hypotheses. Each weak hypothesis has a weight associated with it that is updated after each round, based on its performance on the training set. A separate set of weights are used to bias the training set so that the importance of incorrectly classified examples are increased. Thus, the weak learners can focus on them in successive rounds. A linear combination of the weak hypotheses and their weights are used to make a strong hypothesis for classification.

More formally, for each unique 3D gesture pair, our algorithm takes as input training set $(\vec{x}_1, y_1), ..., (\vec{x}_m, y_m)$, where each $\vec{x}_i$, represents a feature vector containing J features. Each y_i labels $\vec{x}_i$ using label set $Y = \{-1, 1\}$, and m is the total number of training samples. Since we are using a pairwise approach, our algorithm needs to train all unique pairs of gestures. For each unique pair, the AdaBoost algorithm is called on a set of weak learners, one for each feature discussed in Section 3.3. We chose this approach because we found, based on empirical observation, that our features can discriminate between different 3D gesture pairs effectively. We wanted the features to be the weak learners rather than having the weak learners act on the features themselves. Thus, each weak learner C_j uses the j^{th} element in the $\vec{x}_i$ training samples, which is noted by $\vec{x}_i(j)$ for $1 \leq j \leq J$.

3.2.1 Weak Learner Formulation

We use weak learners that employ a simple weighted distance metric, breaking $(\vec{x}_1, y_1), ..., (\vec{x}_m, y_m)$ into two parts corresponding to the training samples for each gesture in the gesture pair. Assuming the training samples are consecutive for each gesture, we separate $(\vec{x}_1, y_1), ..., (\vec{x}_m, y_m)$ into $(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)$ and $(\vec{x}_{n+1}, y_{n+1}), ..., (\vec{x}_m, y_m)$ and define $D^1(i)$ for $i = 1, ..., n$ and $D^2(i)$ for $i = n+1, ..., m$ to be training weights for each gesture. Note that in our formulation, D^1 and D^2 are the training weights calculated in the AdaBoost algorithm (see Section 3.2.2).

For each weak learner C_j in each feature vector $\vec{x}_i(j)$ in the training set, the weighted averages are then calculated as

$$\mu_{j1} = \frac{\sum_{k=1}^{n} x_k(j) D^1(k)}{\sum_{l=1}^{n} D^1(l)} \quad (7)$$

and

$$\mu_{j2} = \frac{\sum_{k=n+1}^{m} x_k(j) D^2(k)}{\sum_{l=n+1}^{m} D^2(l)}. \quad (8)$$

These averages are used to generate the weak hypotheses used in the AdaBoost training algorithm. If a given feature value for a candidate gesture is closer to μ_{j1}, the candidate is labeled as a 1, otherwise the candidate is labeled as a -1. If the feature value is an equal distance away from μ_{j1} and μ_{j2}, we simply choose to label the gesture as a 1. Note that it is possible for the results of a particular weak classifier to obtain less than 50% accuracy. If this occurs the weak learner is reversed so that the first gesture receives a -1 and second gesture receives a 1. This reversal lets us use the weak learner's output to the fullest extent.

3.2.2 AdaBoost Algorithm

For each round $t = 1, ..., T * J$ where T is the number of iterations over the J weak learners, the algorithm generates a weak hypothesis $h_t : X \rightarrow \{-1, 1\}$ from weak learner C_j and the training weights $D_t(i)$ where $j = mod(t-1, J) + 1$ and $i = 1, ..., m$. This formulation lets us iterate over the J weak learners and still conform to the AdaBoost framework [14]. Indeed, the AdaBoost formulation allows us to select weak classifiers from different families at different iterations.

Initially, $D_t(i)$ are set equally to $\frac{1}{m}$, where m is the number of training examples for the gesture pair. However, with each iteration the training weights of incorrectly classified examples are increased so the weak learners can focus on them. The strength of a weak hypothesis is measured by its error

$$\varepsilon_t = Pr_{i \sim D_t}[h_t(\vec{x}_i(j)) \neq y_i] = \sum_{i: h_t(\vec{x}_i(j)) \neq y_i} D_t(i). \quad (9)$$

Given a weak hypothesis, the algorithm measures its importance using the parameter

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right). \quad (10)$$

With α_t, the distribution D_t is updated using the rule

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(\vec{x}_i(j)))}{Z_t} \quad (11)$$

where Z_t is a normalization factor ensuring that D_{t+1} is a probability distribution. This rule increases the weight of samples misclassified by h_t so that subsequent weak learners will focus on more difficult samples. Once the algorithm has gone through $T * J$ rounds, a final hypothesis

$$H(x) = \text{sgn} \left(\sum_{t=1}^{T*J} \alpha_t h_t(x) \right) \quad (12)$$

is used to classify gestures where α_t is the weight of the weak learner from round t, and h_t is the weak hypothesis from round t. If $H(x)$ is positive, the new gesture is labeled with the first gesture in the pair and if $H(x)$ is negative it is labeled with the second gesture in the pair.

These strong hypotheses are computed for each pairwise recognizer with the labels and strong hypothesis scores tabulated. To combine the results from each strong hypothesis we use the approach suggested by Friedman [5]; the correct classification for the new gesture is simply the one that wins the most pairwise comparisons. If there is a tie, then the raw scores from the strong hypotheses are used and the one of greatest absolute value breaks the tie.

3.3 Feature Set

We based the features used in the linear and AdaBoost classifier on Rubine's feature set [13]. Since Rubine's features were designed for 2D gestures using the mouse or stylus, we extended them to work for 3D gestures. The Wiimote and Wii MotionPlus sense linear acceleration and angular velocity respectively. Although we

could have derived acceleration and angular velocity-specific features for our study, we chose to make an underlying assumption to treat the acceleration and angular velocity data as position information in 3D space. This assumption made it easy to adapt Rubine's feature set to the 3D domain and the derivative information from the Wiimote and Wii MotionPlus.

The first feature in the set, quantifies the total duration of the gesture in milliseconds, followed by features for the maximum, minimum, mean and median values of x,y and z. Next we analyzed the coordinates in 2D space by using the sine and cosine of the starting angle in the XY and the sine of the starting angle in the XZ plane as features. Then the feature set included features with the sine and cosine of the angle from the first to last points in the XY and the sine of the angle from the first to last points in the XZ plane. After that, features for the total angle traversed in the XY and XZ planes, plus the absolute value and squared value of that angle, completed the features set which analyzes a planar surface. Finally, the length of the diagonal of the bounding volume, the Euclidian distance between the first and last points, the total distance traveled by the gesture and the maximum acceleration squared fulfill Rubine's list.

Initially, we used this feature set for both the Wiimote and the Wii MotionPlus[1], totaling 58 features used in the two classifiers. However, after some initial pilot runs, we discovered singular common covariance matrices were forming with the linear classifier. A singular common covariance matrix cause the matrix inverse needed to find the weights for the linear evaluation functions impossible. We found that these singular matrices were formed when trying to use the feature set with the Wii MotionPlus attachment. Because of this problem, we had to cull the MotionPlus feature set to use only the minimum and maximum x, y, and z values, the mean x, y, and z, values, and the median x, y, and z values. Thus, 29 features were used when running the experiments with the Wiimote and 41 features were used when running experiments with the Wiimote coupled with the Wii MotionPlus.

4 Gesture Set

As Section 2 suggests, the majority of the work on 3D gesture recognition with accelerometer and gyroscope-based input devices contain experiments using only four to 10 unique gestures. In order to better understand how many gestures these types of devices can accurately recognize and how many training samples would be needed to do so, we chose to more than double that set and use 25 gestures. The gestures, depicted graphically in Figure 2, are performed by holding the Wiimote in various orientations. For a majority of the gestures, the orientation and rotation are the same for both left and right handed users. The only exceptions are the gestures Tennis Swing, Golf Swing, Parry, and Lasso. The gestures Tennis Swing and Golf Swing are performed on either the left or right side of the user, corresponding to the hand that holds the Wiimote. For the Parry gesture, a left handed person will move the Wiimote towards the right, while a right handed person will move the Wiimote towards the left. Finally, the Lasso gesture requires a left and right handed user to rotate in opposite directions, clockwise versus counterclockwise, respectively.

We developed this gesture set by examining existing video games that make use of 3D spatial interaction, specifically from the Nintendo Wii gaming console. We examined a variety of different games from different genres including sports games, first person shooters, fighting games, and cooking games. Although the gesture set is game specific, it is general enough to work in a wide variety of virtual and augmented reality applications. Initially, we focused on simple movements such as line to the right and line to the left which could be applied to various actions. From there, slightly more complex gestures were added which involved closed figures such as the square, triangle and circle. With this group, we add user variations in velocity, duration and distance traversed. Finally the last set of maneuvers allowed for more freedom in body movement in an effort to help disambiguate gestures during feature analysis. Examples of these gestures include golf swing, whip and lasso.

[1]The Wii MotionPlus used maximum angular velocity squared instead of maximum acceleration squared.

5 3D Gesture Data Collection

Machine learning algorithms such as the linear and AdaBoost classifiers need data to train on. In order for the algorithms to learn and then recognize gestures, we created a gesture database to use in both training and testing. We recruited 17 participants (4 female and 13 male) from the University of Central Florida, of which four men were left handed, to provide the gesture samples. Each participant had experience using the Nintendo Wii gaming console, with two participants doing so on a weekly basis.

Several steps were taken to ensure that participants provided good gestures samples. First, they were given a brief overview of the Wiimote gesture data collection interface and a demonstration of how to interact with the application. After the demonstration, participants were presented with an introduction screen asking them to choose what hand they would hold the Wiimote with to perform the gestures. This decision also told the application to present either left or right-handed gesture demonstration videos. Next, an orientation window (see Figure 3) was displayed to help participants learn how to perform the gestures before data collection began. Each of the 25 gestures are randomly listed on the left hand side of the window to reduce the order effect due to fatigue. Participants moved through the list of gestures using the up and down buttons on the Wiimote. At any time, participants could view a video demonstrating how to perform a gesture by pressing the Wiimote's right button. These videos demonstrated how to hold the Wiimote in the proper orientation in addition to showing how to actually perform the motion of the gesture. Before the gesture collection experiment began, an orientation phase required participants to perform and commit one sample for each gesture. This reduced the order effect due to the learning curve. To perform a gesture, particpants pressed and held down the Wiimote's "B" button to make a gesture. After participants released the "B" button, they could either commit the gesture by pressing the Wiimote's "A" button or redo the gesture by pressing and holding the "B" button and performing the gesture again. This feature enabled participants to redo gesture samples they were not happy with. Once a gesture sample is committed to the database, the system pauses for two seconds, preventing the start of an additional sample. This delay between samples allows participants to reset the Wiimote position for the next gesture sample. In addition, the delay prevents the user from trying to game the system by quickly performing the same gesture with little to no regard of matching the previous samples of that gesture.

After participants created one sample for each of the 25 gestures, they entered data collection mode. The only difference between orientation mode and data collection mode is the amount of samples which need to be committed. When participants commit a training sample for a particular gesture, a counter is decremented and displayed. A message saying the collection process for a given gesture is complete is shown after 20 samples have been entered for a gesture. This message indicates to the participant that they can move on to providing data for a remaining gesture. A total of 20 samples for each gesture, or 500 samples in all, is required for a full user training set. In total, 8,500 gesture samples were collected[2]. Each data collection session lasted from 30-45 minutes.

6 3D Gesture Recognition Experiments

With the 3D gesture database, we were able to run a series of experiments to examine both learning algorithms in terms of user depen-

[2]The gesture database can be downloaded at http://www.eecs.ucf.edu/isuelab/downloads.php.

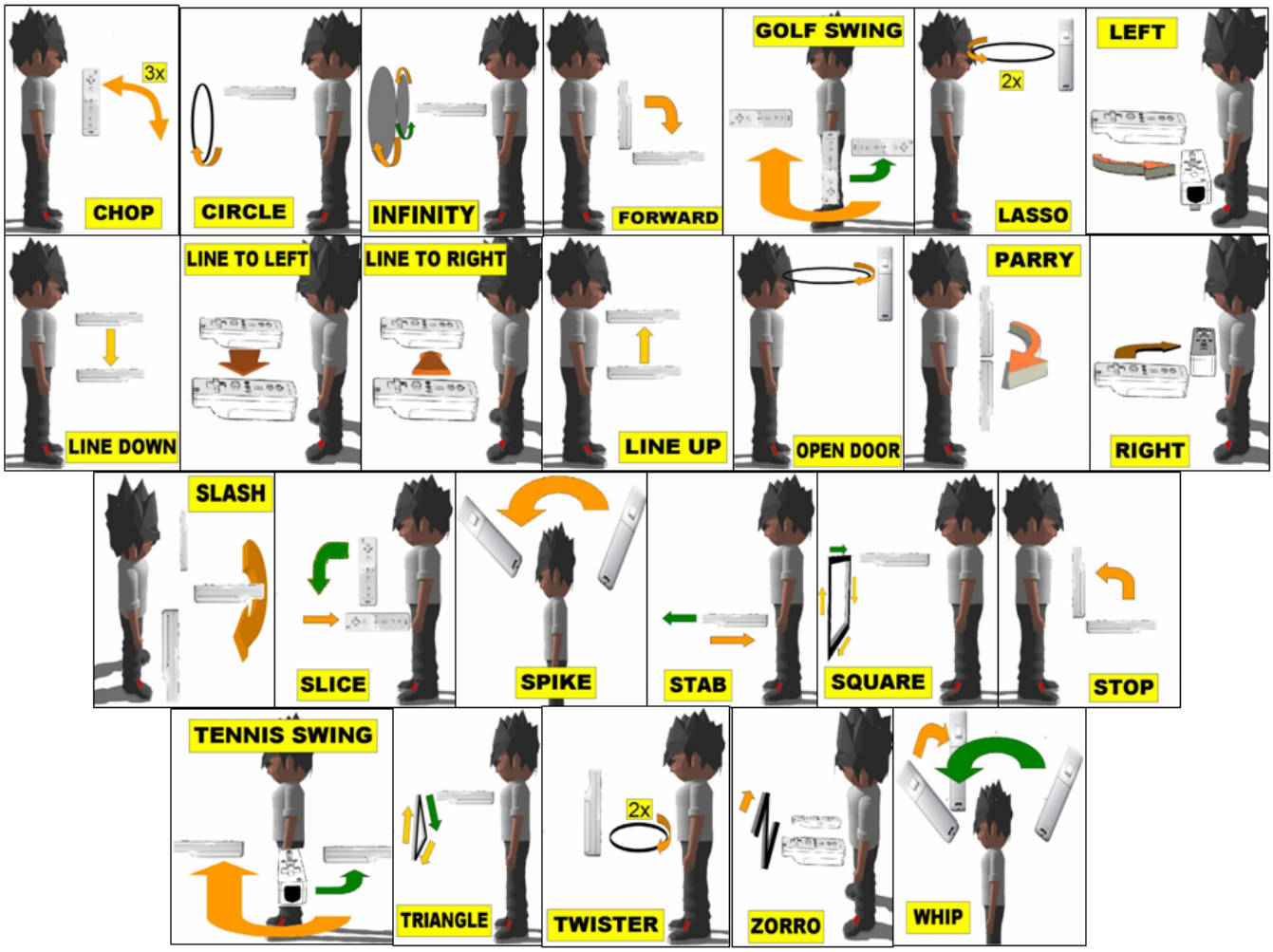

Figure 2: Illustration showing the 25 gestures used in our study are performed with a Wiimote. For compound movements the green arrows show the initial movement and the orange arrows show the remaining movements.

dent and user independent recognition. The primary metric analyzed in our experiments is gesture classification accuracy, while having the over-arching goal of maximizing the number of gestures correctly recognized at varying degrees of training the machine learning algorithms. For both the dependent and independent experiments, the linear and AdaBoost classifiers were tested using the Wiimote data only and using the Wiimote in conjunction with the Wii MotionPlus attachment. Thus, our experiments attempted to answer the following questions for both the user dependent and independent cases:

- How many of the 25 gestures can each classifier recognize with accuracy over 90%?
- How many training samples are needed per gesture to achieve over 90% recognition accuracy?
- How much accuracy improvement is gained from using the Wii MotionPlus device?
- Which classifier performs better?
- Which gestures in our gesture set cause recognition accuracy degradation?

6.1 User Dependent Recognition Results

For the user dependent recognition experiments, we used a subset of samples for each gesture from a single user to train the machine learning algorithms and the remaining samples to test the recognizers. This approach is equivalent to having a user provide samples to the recognizer up front, so the algorithm can be tailored to that particular user. For each user dependent test, two categories of experiments were created: classification over all 25 gestures and finding the maximum recognition accuracy rate over as many gestures as possible. Both categories were then broken into three experiments providing 5, 10, or 15 training samples per gesture with the remaining samples used for testing accuracy. Each experiment was executed on all four of the classifiers mentioned earlier. The experiment set was conducted on each of the 17 participant's data.

The results of trying to recognize all 25 gestures in each experiment are shown in Figure 4. We analyzed this data using a 3 way repeated measures ANOVA and found significance for the number of training samples ($F_{2,15} = 17.47, p < 0.01$), the classification algorithm ($F_{1,16} = 119.42, p < 0.01$), and the use of the Wii MotionPlus data ($F_{1,16} = 8.23, p < 0.05$). To further analyze the data, we also ran pairwise t tests. The most notable observation is the high level of accuracy across all experiments. In particular, the linear classifier gave a mean accuracy value of 93.6% using only 5 training

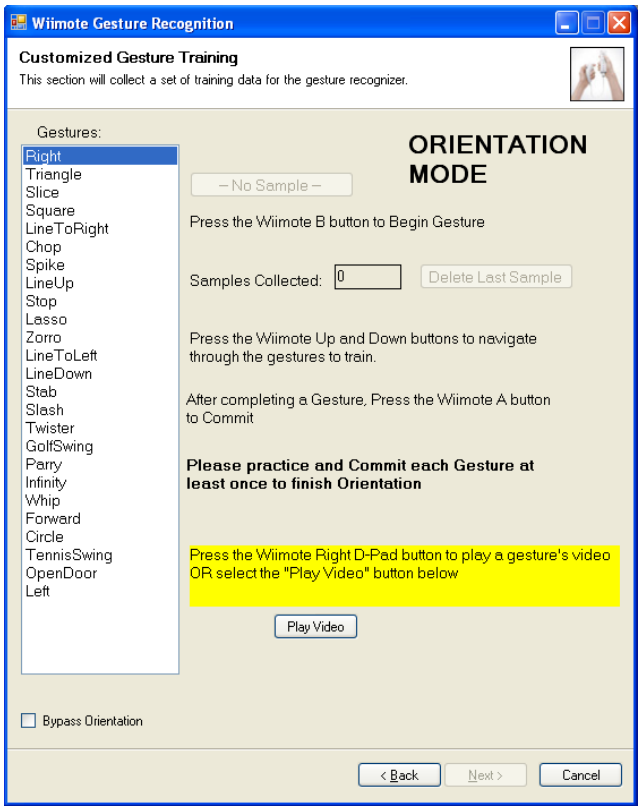

Figure 3: The gesture collection window that each participant interacted with in order to provide 20 samples for each of the the 25 gestures.

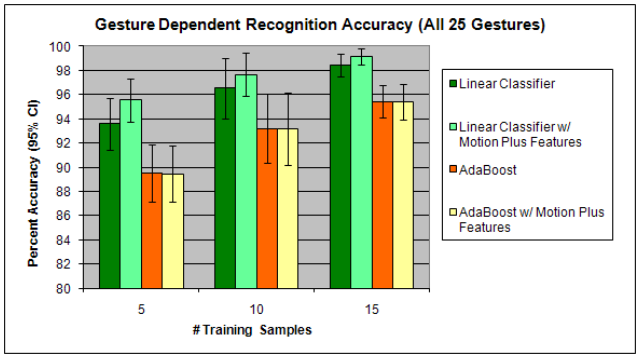

Figure 4: The average recognition results for the user dependent experiments over all 25 gestures. The results are grouped by the number of training samples (5, 10, or 15) provided for each gesture within an experiment. A single user dependent experiment utilized two classifiers (linear and AdaBoost), each executing with either the Wiimote or Wiimote + MotionPlus input device producing four accuracy values.

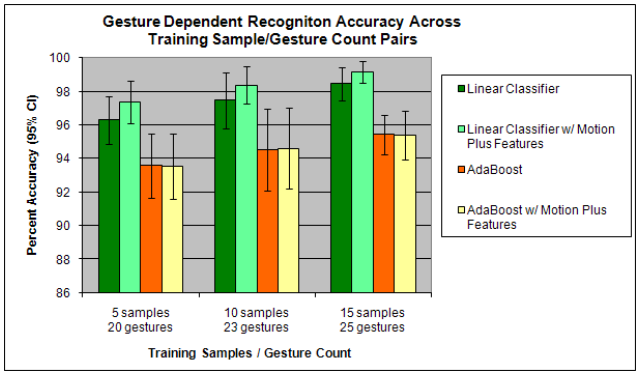

Figure 5: The average recognition results for the user dependent experiments over a varying number of gestures. The goal of this experiment was to recognize, with high accuracy, as many gestures as possible with different levels of training. The results are grouped by the number of training samples (5, 10, or 15) provided for each gesture within an experiment. A single user dependent experiment utilized two classifiers (linear and AdaBoost), each executing with either the Wiimote or Wiimote + MotionPlus input device producing four accuracy values.

samples for each gesture and 98.5% using 15 training samples per gesture ($t_{16} = -5.78, p < 0.01$). Furthermore, the linear classifier outperformed AdaBoost in every situation by at least 3% (see Table 1 for the statistical results). Another important result shown in Figure 4 is the role of the Wii MotionPlus in both recognition algorithms. The Wii MotionPlus significantly increased the recognition accuracy for the linear classifier to 95.5% using 5 training samples ($t_{16} = -2.81, p < 0.05$) per gesture and 99.2% using 15 training samples per gesture ($t_{16} = -2.54, p < 0.05$). For AdaBoost, the change in accuracy was negligible.

While these accuracy levels are good for some applications, we wanted to know how many and even more importantly, which gestures, would need to be removed from the classification set in order to improve recognition results. To begin this examination, the set of gestures frequently recognized incorrectly from the previous three experiment categories were removed. Within this set we noticed a few gestures with similar attributes such as Tennis Swing and Golf Swing or Square and Triangle. These gesture pairs involve similar movement which caused the classifiers to misidentify each type as the other. Once the poorly classified gestures were extracted the accuracy results easily increased to above 98%. We systematically added each of the removed gestures back into gesture set on a case by case basis, leaving one gesture from each similar pairing out in order to improve the recognition on the other included gesture from each pair. The results are shown in Figure 5. As with the previous experiment, we ran a 3 way repeated measures ANOVA as well as t tests and found significant results for the training sample/number of gestures pairs ($F_{2,15} = 5.01, p < 0.05$), the classification algorithm ($F_{1,16} = 48.55, p < 0.01$), and the use of the Wii MotionPlus data ($F_{1,16} = 9.69, p < 0.01$).

In the experiment using 5 training samples, the gestures Forward, Golf Swing, Spike, Triangle and Line to Left were removed, thereby producing over 93.5% accuracy for AdaBoost and over 96.3% for the linear classifier. Once the number of training samples was increased to 10, the set of removed gestures included only Spike and Triangle. This experiment yielded higher accuracy with the linear classifier ($t_{16} = -1.90, p = 0.075$) and AdaBoost ($t_{16} = -1.07, p = 0.3$), but these results were not significant. Finally, since the recognition rates for 15 training samples were already greater than 95%, no gestures were removed during this last test. These results show that recognition accuracy rates as high as 97% can be achieved for 20 gestures using only 5 samples per gesture and 98% for 23 gestures using 10 samples per gesture with the Wiimote coupled with the Wii MotionPlus attachment. In fact, for the linear classifier, the recognizer obtained significantly higher accuracy using the Wii MotionPlus attachment in the 5 training sample/20 gesture case ($t_{16} = -2.32, p < 0.05$) and the 10 training sample/23 gesture case ($t_{16} = -2.18, p < 0.05$).

Comparison	Test Statistic	P Value
Linear5 - Ada5	$t_{16} = 7.17$	$p < 0.01$
LinearMP5 - AdaMP5	$t_{16} = 8.36$	$p < 0.01$
Linear10 - Ada10	$t_{16} = 6.48$	$p < 0.01$
LinearMP10 - AdaMP10	$t_{16} = 6.54$	$p < 0.01$
Linear15 - Ada15	$t_{16} = 7.69$	$p < 0.01$
LinearMP15 - AdaMP15	$t_{16} = 7.16$	$p < 0.01$

Table 1: Results from a set of t-tests showing significance differences between the linear classifier and AdaBoost indicating the linear classifier outperforms AdaBoost in our test cases. Note that under the comparison column, MP stands for whether the Wii MotionPlus was used and the number represents how many samples were used for training the algorithms.

6.2 User Independent Recognition Results

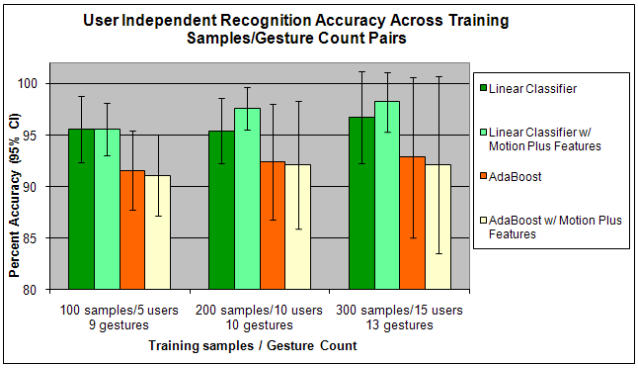

Figure 6: The average recognition results for the user independent experiments over a varying number of gestures. The goal of this experiment was to recognize, with high accuracy, as many gestures as possible when given different levels of training. The results are grouped by the number of user training samples (100,200, or 300) per gesture used for training within an experiment. These numbers are analogous to using 5, 10, and 15 users' data for training. A single user independent experiment utilized two classifiers (linear and AdaBoost), each executing with either the Wiimote or Wiimote + MotionPlus input device producing four accuracy values.

For the user independent recognition experiments, we used a subset of the 17 user study gesture data files for training. From the remaining files, recognition is performed and reported on a per user basis. This approach is equivalent to having the recognizers trained on a gesture database and having new users simply walk up and use the system. The benefit of this approach is there is no pre-defined training needed for each user at a potential cost in recognition accuracy. We ran three tests in this experiment, using data from 5, 10, and 15 users from our gesture database for training with the remaining user data for testing. As in the user dependent study, each scenario was executed on all four of the classifiers mentioned earlier. Note that for the independent recognition results, we did not perform any statistical analysis on the data because we did not have an equal number of samples for each condition and the sample size for the 15 user case as two small (only two test samples). As part of future work, we plan to perform cross validation on the data to deal with this problem so proper statistical comparisons can be made.

The results, shown in Figure 6, shows that the linear classifier outperforms AdaBoost by at least 3% and using the Wii MotionPlus attachment improved recognition for only the linear classifier. However, unlike in the user dependent tests, the Wii MotionPlus slightly hindered AdaBoost accuracy. After removing the poorly classified gestures, we followed the reintroduction method we used for the user dependent tests. The linear classifier was able to recognize a total of 9, 10, and 13 gestures with mean accuracy values of 95.6%, 97.6%, and 98.3% respectively using the Wiimote coupled with the Wii MotionPlus attachment. The gestures enabled for the 5 user training set included Forward, Stop, Open Door, Parry, Chop, Circle, Line to Right, Line Up and Stab. When using the 10 user training set, the Twister and Square gestures were added while maintaining slightly higher accuracy levels. On the other hand, the gesture Open Door was removed because the newly introduced training data increased confusion among samples of that type. Finally, for the 15 user training set, the gestures Open Door (again), Infinity, Zorro and Line Down were added but the Twister gesture was removed.

7 Discussion and Future Work

From our experiments, we can see that 25 3D gestures can be recognized using the Wiimote coupled with the Wii MotionPlus attachment at over 99% accuracy in the user dependent case using 15 training samples per gesture. This result significantly improves upon the results in the existing literature in terms of the total number of gestures that can be accurately recognized using a spatially convenient input device. There is, of course, a tradeoff between accuracy and the amount of time needed to enter training samples in the user dependent case. 15 training samples per gesture might be too time consuming for a particular application. As an alternative, the results for 5 training samples per gesture only shows a small accuracy degradation. For the user independent case, we can see the accuracy improves and the number of gestures that can reliably be recognized also increases as the number of training samples increases. Although the overall recognition accuracy and the number of gestures was higher in the dependent case (as expected), the independent recognizer still provides strong accuracy results.

Comparing the two classifiers in the experiments shows the linear classifier consistently outperforms the AdaBoost classifier. This is somewhat counterintuitive given the AdaBoost classifier is a more sophisticated technique. However, as we discussed in Section 3.3, the linear classifier can suffer from the singular matrix problem which can limit its utility when new features that could improve accuracy are added.

With the Wii MotionPlus, we also tried to adjust for gyroscopic drift in data. Our attempt at calibration involved setting the Wiimote coupled with the MotionPlus device on a table with the buttons down for approximately 10 seconds, followed by storing a snap shot of the roll, pitch and yaw values. That snapshot was then used as a point of reference for future collection points. However, using those offsets caused decreased accuracy in AdaBoost and singular matrices in the linear classifier leading to the use of raw gyroscope data instead.

The results from our experiments show there are other interesting areas for future work. First, it would be interesting to see if more than 25 gestures could be reliably recognized using the Wiimote and Wii MotionPlus attachment and how many training samples would be needed, in the user dependent and independent cases, to achieve similar accuracy values reported in our current experiments. Second, for our user independent case, it is still unclear how far we can go toward increasing the number of gestures recognized at accuracy levels in the 95 to 98% range. It would be interesting to increase the amount of training data from more users to determine when an accuracy fall off would occur. Third, given the problems we encountered with the Wii MotionPlus device, determining the most appropriate calibration method could lead to even higher accuracy numbers. Finally, it is still unclear why the AdaBoost classifier did not perform as well as the linear classifier across our experiments. Thus, it would be interesting to explore the two algorithms further to determine what the cause is for the AdaBoost

classifier to have inferior performance.

8 Conclusion

We have presented a systematic study on 3D gesture recognition with spatially convenient input devices. In particular, we examined the linear acceleration sensing Nintendo Wii Remote and the angular velocity sensing Wii MotionPlus to determine how many gestures could be recognized with a high degree of accuracy We created a 3D gesture database, collecting data on 25 distinct gestures totalling 8500 gestures samples and used this data to compare two machine learning algorithms, a linear and AdaBoost classifier, varying the number of gestures samples use to train them. We examined both user dependent and user independent recognition configurations and found that in the user dependent case, all 25 gestures could be recognized at over 99% accuracy with the linear classifier, using 15 training samples per gesture with the Wii Remote coupled with the Wii MotionPlus. We also found that this combination could recognize up to 13 gestures at over 98% accuracy in the user independent case using 300 training samples per gesture. Although our work focused on the Wiimote input device, we believe our results are applicable to other accelerometer and gyroscope-based input devices such as cell phones. We also believe that there is still more work to do to improve the number of gestures and the recognition accuracy of spatially convenient devices.

Acknowledgements

This work is supported in part by NSF CAREER award IIS-0845921 and NSF Award IIS-0856045. We wish to thank the anonymous reviewers for their valuable suggestions.

References

[1] K. Beedkar and D. Shah. Accelerometer based gesture recognition for real time applications, http://www.cc.gatech.edu/grads/d/dhanik/project/Accel.pdf, 2008.

[2] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[3] G. Caridakis, K. Karpouzis, C. Pateritsas, A. Drosopoulos, A. Stafylopatis, and S. Kollias. Hand trajectory-based gesture recognition using self-organizing feature maps and markov models. 2008 IEEE International Conference on Multimedia & Expo (ICME), 2008.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, 2001.

[5] J. H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996.

[6] S. Kallio, J. Kela, and J. Mantyjarvi. Online gesture recognition system for mobile interaction. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, pages 2070– 2076. IEEE, 2003.

[7] L. Kratz, M. Smith, and F. J. Lee. Wiizards: 3d gesture recognition for game play input. In *Future Play '07: Proceedings of the 2007 conference on Future Play*, pages 209–212, New York, NY, USA, 2007. ACM.

[8] J. J. LaViola and R. C. Zeleznik. A practical approach for writer-dependent symbol recognition using a writer-independent symbol recognizer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1917–1926, 2007.

[9] J. Mäntyjärvi, J. Kela, P. Korpipää, and S. Kallio. Enabling fast and effortless customisation in accelerometer based gesture interaction. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 25–31, New York, NY, USA, 2004. ACM.

[10] S. Perrin, Á. Cassinelli, and M. Ishikawa. Gesture recognition using laser-based tracking system. In *FGR*, pages 541–546, 2004.

[11] T. Pylvänäinen. Accelerometer based gesture recognition using continuous hmms. *Pattern Recognition and Image Analysis*, pages 639–646, 2005.

[12] M. Rehm, N. Bee, and E. André. Wave like an egyptian: accelerometer based gesture recognition for culture specific interactions. In *BCS-HCI '08: Proceedings of the 22nd British HCI Group Annual Conference on HCI 2008*, pages 13–22, Swinton, UK, UK, 2008. British Computer Society.

[13] D. Rubine. Specifying gestures by example. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 329–337, New York, NY, USA, 1991. ACM.

[14] R. E. Schapire. A brief introduction to boosting. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[15] T. Schlömer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a wii controller. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 11–14, New York, NY, USA, 2008. ACM.

[16] M. Turk. Gesture recognition. *Handbook of Virtual Environments*, pages 223–238, 2001.

[17] M. Vafadar and A. Behrad. Human hand gesture recognition using motion orientation histogram for interaction of handicapped persons with computer. In *ICISP '08: Proceedings of the 3rd international conference on Image and Signal Processing*, pages 378–385, Berlin, Heidelberg, 2008. Springer-Verlag.

[18] C. Wingrave, B. Williamson, P. Varcholik, J. Rose, A. Miller, E. Charbonneau, J. Bott, and J. LaViola. Wii remote and beyond: Using spatially convenient devices for 3duis. *IEEE Computer Graphics and Applications*, 30(2), 2010.