Gesture Recognition: Controlling Presentation Software with Microsoft's Kinect

Todd Taomae
Department of Information and Computer Sciences
University of Hawaii at Manoa
Honolulu, Hawaii
Email: ttaomae@hawaii.edu

Robert Ward
Department of Information and Computer Sciences
University of Hawaii at Manoa
Honolulu, Hawaii
Email: rward@hawaii.edu

***Abstract*—Recent advances in technology have created the opportunity for new approaches to computer interaction. Gesture recognition is one of these emerging forms of computer interaction and the potential uses are varied. Some examples are sign language recognition, socially assistive robotics, directional indication through pointing, control through facial gestures, alternative computer interfaces, immersive game technology and affective computing. In previous studies gestures were used to control a primary activity. In this study the control is in support of the primary activity. To investigate the use of gesture control in a secondary support role we have created a program that uses the Microsoft Kinect and the associated libraries to allow a user to control presentation software with gestures. In this support role gestures must be carefully created to not interfere with or limit the primary activity. At the same time the gestures must not create false positives based on gestures used for the primary activity which in this case is giving a presentation. Five control gestures, forward one slide, backward one slide, forward five slides, backward five slides, and a single gesture to start or stop the presentation, were designed and a program was created to detect them. Based on our observations during the development process we created guidelines for gestures that are secondary to the primary goal of the users. The gestures were tested for accuracy of detection and the gestures usability using a group of approximately forty users. The results were used to evaluate the value of the guidelines created for their design.**

I. Introduction

New technologies and tools have allowed gesture recognition to become common both in the business world and in research. Gesture Recognition has a great number of potential uses such as sign language recognition, socially assistive robotics, directional indication through pointing, control through facial gestures, alternative computer interfaces, immersive game technology and affective computing. Another area it has great potential in is remote control of computers which will be the focus of this paper.

In November 2010 Microsoft introduced a new controller for its Xbox game console. It became the fastest selling consumer electronics device according to the Guinness Book of World Records after selling 8 million in just two months. While not designed for use with a computer, researchers and hobbyists quickly attempted to unlock its impressive abilities on a standard PC. To their delight they found it relatively easy to set up, gather data with, and create applications that use its abilities.

Microsoft quickly realized there was a potential market for developers and users beyond the Xbox for which it was designed. They created a Kinect software development kit to assist people interested in developing gesture help for Windows 7 in June 2011. In May 2012, Microsoft released the next version of the Kinect for Windows [4].

The Kinect software development kit (SDK) provides access to low-level streams from the depth sensor, color camera sensor, and four-element microphone array. It also has audio processing capabilities such as acoustic noise suppression, echo cancellation, beam formation to identify the current sound source as well as speech recognition. The abilities provided by the SDK that we specifically utilized are its skeletal tracking. It provides the ability to track two people in its field of view and provides joint information for twenty different joints for each person. To investigate the abilities of the Kinect we have created a program that uses the Kinect and the associated libraries to allow a user to control presentation software with gestures. The accuracy and usability of the gestures created were then measured.The approach is different from previous studies in that it looks at gestures used in support of the primary goal of the activity. The presentation is the primary goal of the user and controlling the viewer supports and is secondary to that activity.

We created five control gestures: forward one slide, backward one slide, forward five slides backward five slides, and a single gesture to start or stop the presentation. These gesture were detected using the Kinect hardware and the Kinect SDK algorithmically. To evaluate both the detection of the gesture and their usability we ran tests with approximately forty different users. Based on our observations during the development process we created development guidelines for gestures that are secondary to the primary goal of the users.

II. Background

A. Related Work

Several recent papers investigated the use of the Kinect as an interface for a PC. For example Kam Lai, Janusz Konrad, and Prakash Ishwar used the Kinect to detect hand gestures to control applications. They concluded that hand gestures were usable in controlled environments if they had data about the subjects hands and performed training to improve recognition. Their use of Euclidean distance metric also requires a great deal of storage and is computational expensive [2]. To avoid

this we will use larger skeletal gestures the will not require the same level of controlled environment.

We do not need to do data collection of gestures for training purposes since Gu, Do, Ou and Sheng showed over eighty percent recognition without training [1]. Although training did raise accuracy of gesture recognition to over eighty percent.

Elimination of these complexities will allow us to use a standard laptop to run our software, use the software in real world environments and focus our efforts on understanding what is required to design gestures.

B. Kinect Skeletal Tracking

The Kinect has many features, but the one that we focused on is its skeletal tracking. Although the Kinect has the ability to detect the position of up to six skeletons and track up to two skeletons at once, in developing our gestures we assume that there is a single user in view of the Kinect. If a skeleton is tracked, the Kinect provides the 3D coordinates of 20 "joints." The joints that it tracks are the head, the shoulder center, the spine, the hip center, and the left and right shoulders, elbows, wrists, hands, hips, knees, ankles, and feet. The joints that we are most concerned with are the hand joints, although we do use a number of other joints to assist in our gesture detection.

Each joint can be either tracked or inferred. A tracked joint is one where the Kinect knows the position, while an inferred joint is one where the Kinect is guessing the position. There are several reasons why the Kinect might not know the position of the joint. The obvious reasons are that the joint may be out of frame or occluded. Another reason is related to how the Kinect detects and tracks the skeleton. In the default tracking mode, which we used, the Kinect detects skeletons based on their distance from the background. This works well in most cases, but if, for example, the user is standing against a wall, the Kinect will probably not be able to track them. This is probably not an issue in almost all cases, but there is another, more subtle problem that can occur. If the hand, for example, is too close to the body, the Kinect wont be able to distinguish it from its background, so the Kinect will not be able to track it.

C. Gesture Development

Selecting a good gesture was a key part of developing our gesture recognition software. This is especially true for the forward one slide gesture, which would be performed the most frequently. A good gesture should comfortable and easy to perform. That means that is should not be physically demanding and that it should not be too specific or constrained. Another important consideration is that it is well connected with the action it is meant to perform.

To understand the basic process of creating a gesture we created a very simple proof of concept gesture. This gesture simply looked for a change in the height of the hand from below to above the head. Raising the right hand indicated forward one slide and raising the left hand indicated back one slide. In creating these gestures we were able to gain a basic understanding of how to detect a gesture algorithmically.

Our testing of this gesture led us to guidelines for what made a good gesture for our software. While flawed in many ways it helped us understand the requirements of a usable gesture. It was easy to have a detection when that was not your intention, it was not symmetrical since left and right are not intrinsically opposite, and it became draining to continually raise your hand to move the slide forward. The following list contains the general guidelines we created to help us develop good gestures:

- Distinct from common motions used in primary activity.
- Natural and pleasant to use. Not physically difficult to perform, particularly if it is a common gesture.
- Minimal precision required to perform gesture.
- Easy to remember. Good mapping from gesture to its associated action.
- Related actions, such as forward and backward, should have related gestures.
- Relatively easy to detect.

The final gesture that we selected for the forward slide gesture was a clockwise circle performed with the right hand, with the center of the circle at the height of the spine joint that the Kinect provides, which is approximately at the users waist. We then use the distance from the shoulder center to the right shoulder to approximate half the users body width. The center of the circle is moved to the right by this amount so that the gesture does not need to be performed at the center of the body. This point was chosen to be at a comfortable position so that it does not put unnecessary strain on the user as they perform the gesture multiple times throughout the presentation. An alternative consideration was to center the point around the elbow, but we found that this was not a reliable point since the users elbow may move with their hand as they perform the gesture.

The backward slide gesture is symmetrical to the forward slide gesture. It is a counter-clockwise circle with the right hand, centered around the same point. Selecting a symmetrical gesture makes it easier for the user to remember the gesture and helps to connect the gesture with the action it performs.

For similar reasons, the forward and backward five slides gestures are very similar to their single slide counterparts. We decided that the gesture should be the same, but with a modifier performed with the left hand. One consideration was to have the gesture performed with both hands. However, this method had some drawbacks. We found that it wasnt as easy to perform and using two hands does not correlate well with going forward or backward five slides. The modifier that we selected was to raise the left hand to the side of the head. To aid in remembering the gesture, the user can hold five fingers up.

In selecting the start and stop gesture, we decided that it wasnt as important that the gesture be comfortable since it would be performed very infrequently. We chose a single gesture to represent either action because the user will only ever need to perform one of the actions. If the presentation has not started, the gesture will perform a start action. If the presentation has started, it will perform a stop action. The gesture that we selected was to have both hands to the side

of the head for at least one second. This gesture needed to be fairly unique since a false detections would mean accidentally stopping the presentation.

III. Approaches / Methods

There are several potential approaches to gesture recognition with the Kinect. Three that we considered were artificial neural networks, detection by example, and algorithmic detection. Artificial neural networks are relatively complex to implement, but they are very efficient and allow you to easily implement new gestures. Detection by example uses machine learning techniques to compare the users movements to known forms. It is effective at handling complex gestures and it tends to improve over time. However, it is also difficult to implement and requires training for each gesture. Algorithmic detection is the simplest of the three approaches because there is no up front work. However, adding new gestures is more difficult than the previous methods because a new algorithm must be developed for each gesture. Algorithmic detection has several other limitations and disadvantages. For example, it works well for simple gestures but it can be very difficult to accurately detect complex gestures. Another limitation is that performance will degrade as the number of gestures increase [4]. Despite these disadvantages, it was the best choice for our needs because we are only detecting a few, relatively simple gestures.

A. Detecting the Circle Gesture

The most important gesture to accurately detect is the circle gesture because it is the one that would be used most commonly during a presentation. For several reasons, we decided not to actually detect a circle. First, it would be difficult for users to create an accurate circle with their hand. The gesture should be simple to perform without having to worrying about performing a very specific movement. Second, it would be difficult to detect algorithmically. We would either have to constrain the users movements to fit nicely into our algorithmic definition of what the circle should be, or the code would quickly become very complex by trying to deal with all the ways that a user might try to create a circle.

Our actual approach was to track the right hand as it traveled through four quadrants, centered around a point located at the right of the users waist. In order to complete the circle gesture, the users hand must return to the starting quadrant. Since the forward slide and backward slide gestures are symmetrical, a single algorithm is used to detect the circle and the direction is determined when the circle is complete.

In order to reduce the number of false detections, we needed to add constraints to the gesture. One constraint that was considered was to make the quadrants bounded so that the user must perform the gesture within a certain area. However, we decided that this would be too limiting and would put unnecessary limitations on the user. The final constraints that we put on the gesture was to have the circle start in one of the two upper quadrants and to put a time limit on the gesture. These two constraints were designed to limit false detections where the user lifts their hand through two quadrants then happens to lower their hand through the remaining two quadrants and finally moves their hand into the starting quadrant.

B. Detecting the Five Slide Modifier and Start/Stop Gesture

Both the five slide modifier and the start and stop gesture require us to detect a hand to the side of the head. The simple solution is to just check that the position of the hand at a given time is at a certain height. The problem with this approach is that the Kinect does not always accurately track a joint, so we cannot rely on using the position at a single point in time. To combat this limitation of the Kinect, we make two changes to the naive approach.

First, we define a range of acceptable heights that the hand may be at rather than specifying an exact height. The range that we defined was centered around the head joint provided by the Kinect and has a range above and below that joint, equal to the distance between the head and the center should joint. The second change that we make is to keep a history of positions of the hand and use the average position of that history. This will handle situations where the Kinect may lose tracking for a single frame and incorrectly infer the joint to be somewhere outside of the acceptable range.

Keeping track of the average positions has a second advantage. By keeping a long enough history, you can enforce that the hand must be, on average, in a certain position for certain length of time. This is used for the start and stop gesture where the users hands must be near their head for at least one second before activating the gesture. This prevents false detections when the user happens to raise both their hands at the same time.

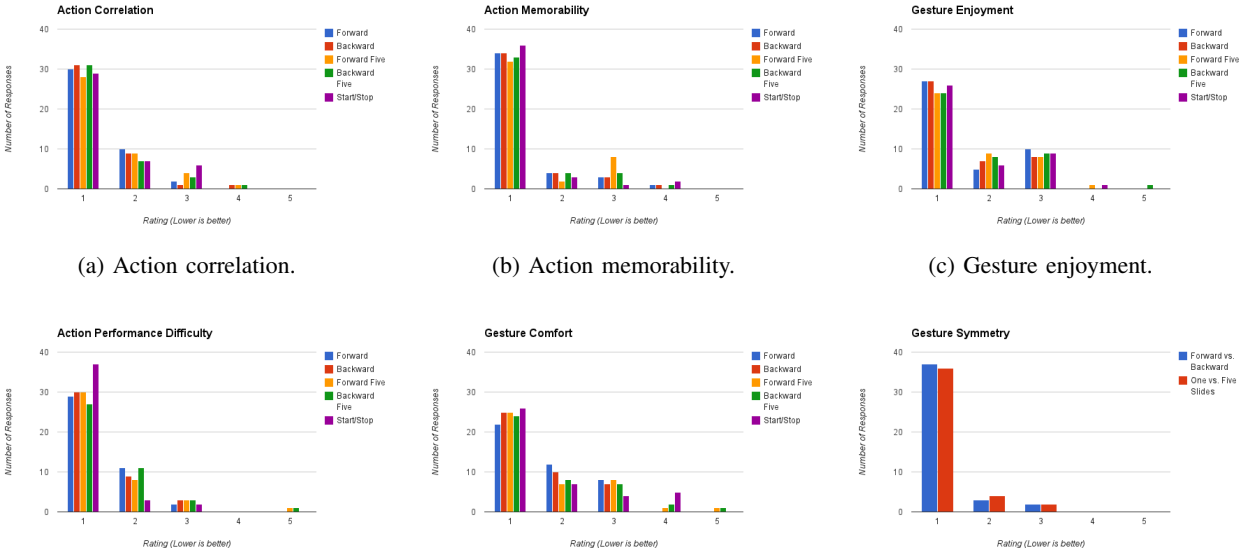
IV. Results

A. Method

Forty two test users were recruited to test the ability of our software to detect the gestures we designed and the ability of users to accurately reproduce the correct gestures. The correct form of gestures were demonstrated to a group of approximately fifteen users. Individually each user was asked to reproduce each gesture for the recognition system. The number of times required for the system to recognize the requested gesture was recorded. If the user need to be reminded of the correct gesture or be given a clarification this was also recorded. After they had completed this part of the testing they were asked out fill out a post test usability survey about their impressions of the gestures.

Our approach requires some control on the test environment and position of the user. The testing was performed in a well lit room. The lighting was not altered and was similar to a normal presentation setting. Users stood directly facing the Kinect at a distance of approximately ten feet.

Testing was designed to examine two general areas, the ability of the software to detect gestures and the usability of the gestures. The goals of our gesture detection testing was to examine the accuracy of the gestures, which we measured as the percentage of time that the gesture recognized. The goals of the usability testing was to examine the physical ease of gestures, memorability of the gestures, quality of mapping of the gesture to its associated action and the enjoyability of the gesture.



(a) Action correlation. (b) Action memorability. (c) Gesture enjoyment.

(d) Action performance difficulty. (e) Gesture comfort. (f) Gesture symmetry.

Fig. 1: Usability survey results

B. Gesture Detection Rates

Table I shows a summary the results of our accuracy testing. The first column is the gesture that we asked the user to perform where "(3)" indicates that the user was asked to perform the gesture three times in succession.

There are two potential ways detection can fail. Either he user does not perform the correct gesture or the software fails to detect a gesture the matches the required gestures. Our approach does not attempt to separate these two types of failure but the gestures were tested prior to user testing and they were recognized consistently. From an overall quality perspective it does not matter which reason the detection failed.

Other than start and stop the gesture all include the same clockwise or counter clockwise gesture. This connected all the gestures and made them easier to remember. In our testing we altered which gesture was requested first to minimize ordering bias. One group was asked to go backward first the other two groups were asked to go forward first. However if you look at the data you will notice that forward has the lowest accuracy rate. To fully remove ordering bias we should have altered the order with every other user since we did not have an even number of groups. The ordering bias does seem to indicate that the gestures are learnable. After only a few reminders of the earlier gestures and a few attempts they achieved a higher detection rate.

Gesture	Average attempts	% accuracy	% reminder/clarification
Start	1.037	96	13
Forward	1.453	69	52
Forward (3)	3.377	89	0
Backward	1.151	87	13
Backward (3)	3.415	88	0
Forward Five	1.302	77	30
Forward Five (3)	3.509	85	0
Backward Five	1.283	78	0
Backward Five (3)	3.642	82	0
Stop	1.094	91	16

TABLE I: Accuracy of gesture detection.

To be viable for actual use it would be necessary to get these rate into the upper ninety percent. Based on these early results that does seems possible. A more robust detection algorithm combined with a longer period of time using the gestures would improve these results.

C. Usability

The post test survey was used to get user feedback on the gestures. A survey or interview after less controlled use of software as well as observation of users attempting to use the software for an actual presentation would result in more accurate feedback. While this is an imperfect way to gather this information given our time constraints this was better than gaining no insight into usability.

Fig. 1 provides a summary of the survey results. The survey used a Likert scale, where a lower number represents a more favorable response. In general the results for all gestures were very similar.

Fig. 1a shows whether users felt that the gestures corresponded with the action it activated in the software. The results indicate that in general the user felt the gestures made sense in terms of the action it activates in the presentation software.

In Fig. 1b we see the feedback about whether users felt they could easily remember the action. Normally data would be collected on whether users actual remember the correct action. The results demonstrate that the user felt that they would be able to remember the gesture and the task it activates in the presentation software. The week after the test we informally asked the testers if they recalled the gestures for each action. Many of them did recall the gestures and corresponding action.

Fig. 1c the feedback on the user enjoyment of performing each gesture. While enjoyment is not always analyzed, people are more likely to use software if it is fun. While these results are positive, the enjoyment metric garnered the least favorable user feedback. It is not clear what might be done to make the gestures more fun. It may be vital for gesture recognition for their use to be enjoyable.

The results shown in Fig. 1d indicate that the users felt the gestures were easy to perform. The inability to recreate the gesture would certainly lead to frustration and make it the software less effective for users.

Fig. 1e summarizes the user feedback on whether the gesture was comfortable. If a gesture must be done several times it is vital that it cause no physical discomfort. The results for all gestures are similar. However the comfort level for forward is the lowest. The forward gesture will be the most common gesture used so it is important that the reason for the result is investigated further and improved. It is possible that the added constraint of starting at the top of the circle decreased its overall comfort level. Further research is necessary to determine the source of this result and to find a possible solution to improve the comfort level of the forward gesture.

Fig. 1f shows the feedback on gestures symmetry which summarizes the users' impression of the relationship between corresponding gestures. The feedback clearly indicates that users saw a clear connection between gestures. The forward and backward gestures were seen as opposites and connected well.

In addition to the quantitative questions, the survey also asked users for comments on each gesture. There were several comments that appeared more than once. Several users mentioned they didnt like the starting at the top of forward gesture. This was a constraint we added to reduce false detections. As is common in software development there is a trade off between these two goals. However it may be possible to revisit this issue once we have improved the overall quality of recognition.

Many users also did not like raising their hands to the side of their head for the start stop gesture, describing it as awkward. They also did not like having to hold this position for several seconds. Again this was a constraint we added to decrease potential false detections. In this case it may be better to accept this less than perfect solution since users will rarely use this gesture.

D. Observations

On top of the test data and survey results, we made several informal observations of the users and their behavior as they interact with the software. We noticed that the users appear to quickly create an internal model of how the gesture detection works. This can be interpreted in several different ways. One interpretation is that the instructions were not clear enough and it should be made more explicit how to perform each gesture. However, an opposing view might be that the software should work with slightly different interpretations of how the gesture should be performed and without needing to give a detailed explanation of each gesture.

There were three observations that we made which seemed to indicate that the users were creating their own internal model, which was often consistent across many users. First, several users, on their first attempt of the circle gesture, would make very slow and deliberate circles, probably in an attempt to ensure that they make an accurate circle for the software to detect. However, this actually has a negative impact on the softwares ability to detect the gesture because of the time limit that was put on the gesture. Another common behavior that we observed in many users was that if their first attempt to activate the circle gesture did not work, they would try to make very large circles. This was particularly problematic when the user attempts to perform the five slides gestures. We found that users would occasionally activate the start and stop gesture because the large circle would reach the height necessary to be considered near the head. If the tests were to be repeated we would be sure to record data on how often this occurs. The last common behavior that we observed was that nearly all users would stop between circles when asked to perform multiple circle gestures in a row.

In addition to the survey results indicating that the users thought the gestures relate well to the corresponding actions and that the forward and backward slide gestures are symmetrical, our observations support these results. After demonstrating that the forward gesture is a clockwise circle, many users seemed to immediately know that the backward gesture would be a counter-clockwise circle. Another observation was that when asked to perform the gesture, users rarely needed to be reminded which direction corresponds to which action.

V. Discussion

Overall our results are positive and promising. Our detections rates are fairly high and seem to improve as the user becomes more comfortable and familiar with the gestures. The gestures are also generally well received by the users, both in terms of usability and relation to the actions they perform. However, there are some obvious weaknesses and limitations in our approach to both the detection and testing.

One limitation of our current detection method is its simplicity. We currently do not use the depth information provided by the Kinect, so there is still room to make the gestures more robust. So far, we have been successful with the algorithmic approach because we have a fairly simple set of gestures. However, if we wanted to add more complex gestures, our current approach would not be ideal. Even our simple gestures became difficult to update as new requirements were added.

Perhaps the greater limitation of our research is our testing. Our testing methods were not as refined as they should have been. One flaw is that we performed all the testing and recorded the data ourselves. This could potentially lead to unintentional biases. The other major flaw is that the testing environment and procedures were not completely consistent across groups or even individual users. One group was tested in one environment with one person performing the demonstration, while the two remaining groups were tested in a different environment with a different person performing the demonstrations. There were no formally specified instructions when asking the users to perform certain gestures, which

means that each user would have slightly different experiences. All of these factors could have introduced errors into our results. Despite these flaws in our methodology, the overall trends were similar across groups, which may indicate that their impact is minimal.

VI. Conclusion

Our proof of concept gesture was very simple and was never intended to be used as one of the gestures but it provided invaluable insights into the requirements of an effective gesture. The early usability test on this gesture being used allowed us to understand its weaknesses and provided insights that allowed us to create guidelines for a good gesture. These guidelines underscore the importance that usability play in the development of gestures. If you can recognize a gestures consistently that is good but to be truly useful it must be something the expected user of the gesture can use effectively.

The gestures received high usability rating from users. The gestures were also recognized at a high rate although for a usable tool the percentage of recognition would need to be improved. This initial test of the gestures developed indicates there is some merit in our guidelines for creating control gestures in a support role.

In testing gesture recognition there are two potential failure points the user does not perform the correct gesture or the software fails to detect a gesture that matches the required gestures. While we did not specifically do this in our testing it may be necessary to have two stages of testing to help identify the exact problem when a gesture is not recognized. In the first stage an expert who knows exactly how to perform the gesture would test the accuracy of the software in recognizing a gesture correctly. In the second stage user testing could be used to find issues with users ability to correctly do the gestures. Even with a two stage process it is likely that testing the accuracy of recognition of a gesture is so closely linked with usability that they cannot be completely separated.

The tools and software for gesture recognition are becoming more readily available and more powerful. In fact the tools already available are quite extensive and powerful. This will lead to gesture recognition becoming more common while it will not replace other forms of input but will augment them. For example the use of voice recognition in conjunction with gesture recognition will be a powerful combination due to its ability to provide context to a user's desires.

VII. Future Work

There are many areas for future work of this research, but they can be broadly divided into two categories. One approach is to improve the software, either by improving the accuracy or by enhancing it with different features. The second approach for further development and research is to improve and further evaluate the usability of the gestures.

We have considered several ways in which we may be able to improve the accuracy of our gesture recognition. The simpler of the two options is to better utilize the 3D information that the Kinect provides. Currently, we do not use the depth information in our gesture recognition. As a result, the gesture recognition is most accurate when the user is directly facing the Kinect. However, if we were to use depth information we can improve the accuracy in situations where the users body is not directly facing the Kinect. To improve our recognition we could also use one or more of the joint filtering techniques discussed in [3]. Some of the possible filter techniques include moving average, double moving average, exponential filter, double exponential filter, and Savitzky-Golay filters. We also may attempt to use body kinematics to improve recognition. This techniques uses physical characteristics of joint movement such as speed and hinge joint limitation to improve recognition [3]. Another, more general approach to improving our gesture recognition is to use a non-algorithmic approach. Although the algorithmic approach has worked well for us so far, there are some obvious limitations. Both artificial neural networks and detection by example tend to have better accuracy, allow for more complex gestures, and allows gestures to be added more easily than with algorithmic detection..

There are also several areas for enhancement of the software that are related to its usability and features rather than its accuracy. One obvious improvement is to allow the user to use either hand to perform each gesture. Another improvement might be to allow multiple users to be in view of the Kinect at once. We could also incorporate voice or other forms of control to improve the users ability to interact with the presentation software. For example performing the gesture while saying a number could be used instead of performing a modifier to advance a certain number of slides.

Perhaps the more interesting direction for improving our software is to improve our user testing and the overall usability. There were some flaws and limitations in our user testing methods. For example, as we mentioned earlier, there appears to be some bias in the accuracy results caused by the order in which we asked users to perform each gesture.

Another limitation of the testing is that it does not test how well the users would perform the gestures in their intended setting. Our tests were performed in a controlled environment, where the user is solely trying to perform the gestures. In a more realistic environment, there would perhaps be uncontrollable external factors and the user would likely be moving both their body and arms, which could lead to lower accuracy and false detections. Our test also only provides a snapshot of usability information. It would be preferable to perform a longitudinal study where we revisit users to see if the gestures are memorable and if they would be able to perform the gestures more accurately with practice.

References

[1] Gu, Y., Do, H., Ou, Y., & Sheng, W. (2012). Human gesture recognition throug a Kinect sensor *IEEE International Converence on Robotics and Biomimetics* (pp. 1379-1384). IEEE.

[2] Lai, K., Konrad, J., & Ishwar, P. (2012). A gesture-driven computer interface using Kinect. *Southwest Symposium on In Image Analysis and Interpretation* (pp. 185-188). IEEE.

[3] Microsoft. (2013, October 1). *Skeletal Joint Smoothing White Paper*. Retrieved from http://msdn.microsoft.com/en-us/library/jj131429.aspx

[4] Webb, J., & Ashley, J. (2012). *Beginning Kinect Programming with the Microsoft Kinect SDK*. Apress.