SpeechBuddy
Presentation Practice and Feedback System

Erica Du
ericadu@mit.edu

May 14, 2015

Abstract

Although public speaking is a very important skill, there are few easily accessible tools to help people get concrete feedback on presentation skills. This paper presents the design and implementation of an offline presentation feedback system called SpeechBuddy, capable of detecting specific body language errors. From a live presentation given to a Kinect connected to a laptop with a webcam, SpeechBuddy will generate a visual report in a web browser, with instances and time stamps of predefined body language features, as well as an embedded mp4 video for the user to review.

1 Introduction

1.1 Motivation

Public speaking is a necessary and useful skill in both professional and personal settings. However, according to a recent poll by the Washington Post, over 25% of Americans fear public speaking [1]. The best way to overcome this fear is through practice and getting feedback on practice runs [2]. Unfortunately, current tools are old-fashioned, limited to grabbing a friend, resulting in skewed feedback, or talking at yourself in a mirror, resulting in no external feedback. Both of these techniques will rarely give you the coaching you would like to improve, as neither of these entities necessarily have professional experience judging presentations, like a teaching assistant for a presentation based class or a speech coach. However, acquiring time with a more experienced set of people to practice is difficult (i.e. limited amount of office hours) or costly. This scenario of wanting to easily and instantly obtain useful and actionable feedback on presentations is the motivation for SpeechBuddy. Ideally, this system would provide feedback for both verbal and nonverbal presentation skills. Currently, SpeechBuddy only supports nonverbal behavior as input. More specifically, SpeechBuddy was built to keep track of at what times and for how long key unfavorable characteristics in body language occur during presentations.

1.2 Use Case

A concrete situation in which this tool may be used is helping 6.UAT students practice for their Previous Project Talks. 6.UAT is a communications and presentation based course that every Course 6 student at MIT is required to take [3]. Because the Previous Project Talk is the very first talk, students have never received a round of feedback from their recitation instructors or TAs, and therefore are often unclear of what is expected of them. In this situation, SpeechBuddy could have the potential to intervene before a students' first talk and could be used to provide them with some degree of feedback on body language during presentations.

1.3 Previous Work

Computer based tools have already been used in various related domains to facilitate practice and improvement. The most closely related tool is My Automated Conversation CoaACH, otherwise known as MACH, which was developed out of the MIT Media Lab as a tool to help people improve their interview skills in an easy, accessible, non-embarrassing way [4]. First users would interact with the system by doing a mock interview with an avatar. Afterwards, the system would provide back qualitative and quantitative data, such as concrete advice for improvement. Multimodal interfaces have also been used in rehabilitation or speech therapy, as well as vocabulary acquisition [5].

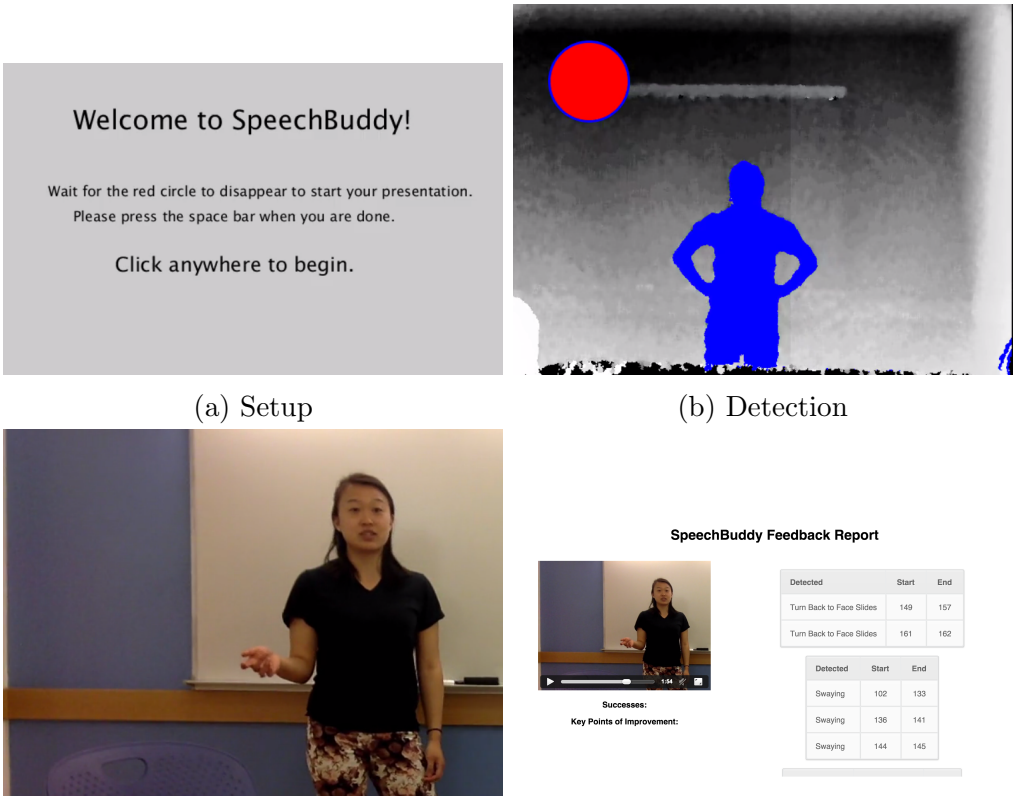
2 Design

SpeechBuddy user flow consists of the following 4 phases, listed sequentially.

1. **Setup**: SpeechBuddy requires the use of both a Kinect and a laptop with a webcam, as well as careful physical setup to perform optimally. The user must position the Kinect and laptop in a manner such that images captured from both devices align and are as similar as possible. A simple command line interface is used to start SpeechBuddy. When a user runs the SpeechBuddy program, a gray pop up box, shown in Figure 1a, appears on screen with a short set of instructions. The use moves onto the next phase after clicking within the popup box.

2. **Detection**: Next, the user must move a reasonable distance away from the Kinect and laptop to a reasonable distance, around 7-9 ft. The user will see a screen with a red circle and rendered depth data, shown in Figure 1b, for the entire duration of this phase.

3. **Presentation**: After the red circle disappears, the view will change from a blue figure against the depth background to a live view from the laptop webcam, which marks the beginning of the Presentation phase. During this phase, the user will give his or her presentation. Upon completion, the user must go to the laptop and press the space key to indicate that he or she is done presenting.



(a) Setup (b) Detection

(c) Presention (d) Review

Figure 1: Phases of User Interaction with SpeechBuddy Interface

4. **Review**: A feedback report is automatically generated from data gathered in the presentation phase, and is automatically displayed via browser in the Review phase. An example of a possible feedback report is shown in Figure 1d. Here, the user can replay portions of the full experience, as well as see every instance of error detection along with time stamps in seconds indicating the start and end time of those errors. Currently, SpeechBuddy supports 7 unique features.

3 Implementation

3.1 Overview

SpeechBuddy, shown in Figure 2 is implemented with 3 key components, a data collector, a data formatter, and a data analyzer, as well as a coordinator. The coordinator is implemented as a bash script, and schedules each task. Data collection and formatting are primarily implemented using the Kinect and the SimpleOpenNI library in Processing. The `ffmpeg` library is also used to compile images into a video file. Lastly, a data analyzer is implemented as a Python script, and processes data files into the final result. The implementation can be found on Github [6].

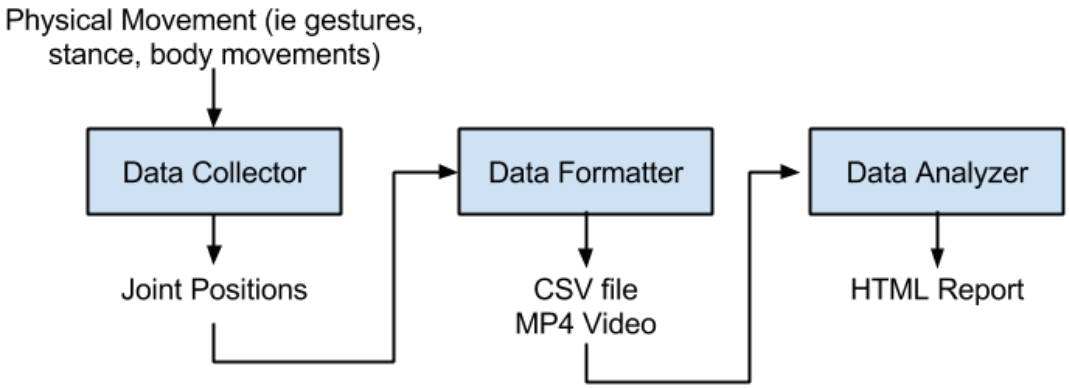


Figure 2: System diagram of SpeechBuddy with each components. Inputs and outputs are also denoted at each stage.

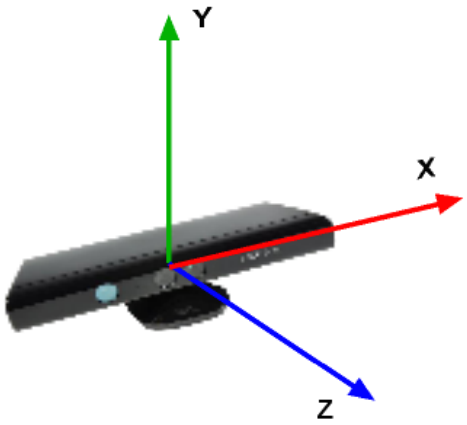


Figure 3: Directionality of axes for Kinect sensor.

3.2 Coordinator

The primary role of the coordinator is to connect the different programs into one cohesive system, as well as schedule and launch each program one at a time, beginning with the data collector. The coordinator is the first part of the system, and receives command line input.

3.3 Data Collection

The Data Collector is written using Processing, and utilizes the SimpleOpenNI library for the Kinect 1473. SpeechBuddy collects all possible skeletal position data of the user from the Kinect, as well as image data from the laptop webcam.

Skeletal Position SpeechBuddy saves all Kinect data in a `positions` table. This includes

Body Part	Name
Head	SimpleOpenNI.SKEL_HEAD
Neck	SimpleOpenNI.SKEL_NECK
Left Shoulder	SimpleOpenNI.SKEL_LEFT_SHOULDER
Left Elbow	SimpleOpenNI.SKEL_LEFT_ELBOW
Left Hand	SimpleOpenNI.SKEL_LEFT_HAND
Right Shoulder	SimpleOpenNI.SKEL_RIGHT_SHOULDER
Right Elbow	SimpleOpenNI.SKEL_RIGHT_ELBOW
Right Hand	SimpleOpenNI.SKEL_RIGHT_HAND
Torso	SimpleOpenNI.SKEL_TORSO
Left Hip	SimpleOpenNI.SKEL_LEFT_HIP
Left Knee	SimpleOpenNI.SKEL_LEFT_KNEE
Left Foot	SimpleOpenNI.SKEL_LEFT_FOOT
Right Hip	SimpleOpenNI.SKEL_RIGHT_HIP
Right Knee	SimpleOpenNI.SKEL_RIGHT_KNEE
Right Foot	SimpleOpenNI.SKEL_RIGHT_FOOT
Center of Mass	SimpleOpenNI.getCoM({{ userID }}, com)
Time From Program Start	millis()
User ID	first element of SimpleOpenNI.getUsers()

Table 1: A model of data gathered in positions table. Each body part is actually saved in three separate cells, one for each of the three coordinates in space.

all 15 possible joint positions from the SimpleOpenNI library, as well as center of mass information. Each row in the table includes a user ID, the timestamp, and a column for each coordinate for each part. Note that the data collector saves the `x` coordinate, the `y` coordinate, and the `z` coordinate of each of these parts separately. A more comprehensive look at the data collected from the Kinect is shown in Table 1.

Webcam Images From launch, SpeechBuddy saves TIF images for each frame from the webcam. These are all saved sequentially in a newly created `images` directory.

3.4 Data Formatting

In this stage, SpeechBuddy takes raw collected data and saves it into a usable format in preparation for processing. SpeechBuddy takes the `positions` table at the end of the presentation and saves the table as a CSV file. In addition, SpeechBuddy takes the TIF images created from the webcam and transforms them into a mp4 video.

3.5 Data Processing

SpeechBuddy maps raw position data into possible body movements by performing the following operations and utilizing the following temporary data structures:

1. Input: CSV file

2. Converts CSV file into usable form by creating a dictionary object mapping column name of CSV file to list of saved spatial coordinates (cells of that column) in temporal order. This dictionary object is called `positions`.

   `positions = { COL_NAME: [` $coordinate_0$`,` $coordinate_1$`, ... ,` $coordinate_n$ `], ... }`

3. Prunes `positions` by cutting some number of elements and the beginning and end of each list to account for time it takes to approach and step away from the laptop. Currently, the parameters `START_BUFFER` and `END_BUFFER` refer to the number of elements SpeechBuddy will prune. Currently, `START_BUFFER` is set to 50, and `END_BUFFER` is set to 100. This new, shortened version of `positions` is referred to as `pos`.

   `pos = { COL_NAME: [` $coordinate_{START}$`, ... ,` $coordinate_{END}$ `], ... }`

4. Instead of analyzing all the user data at once, the data processor divides the `pos` object into `snapshot` objects, which is just the `pos` data structure with the lists shortened into chunks of length `WINDOW`. The averages and variances of various components in these snapshots are then combined and used to make inferences about the behavior of the user during shorter periods of time throughout the presentation. Threshold values and attributes were empirically determined, trained on various instances of the given body language feature. These values can be found in Table 3.

   Each position i in the `pos` corresponds to the beginning of the ith `snapshot`.

   (a) `snapshot` $=$ `{ COL_NAME: [` $coordinate_i$`,` $coordinate_{i+1}$`, ... ,` $coordinate_{i+WINDOW}$ `], ... }`

   (b) **Swaying Detector**: We expect a the head and torso of the user to move in the x direction, and the left and right shoulder to remain at a fairly constant distance away from the camera (not turning towards or away). SpeechBuddy will recognize swaying in the `snapshot` if variance of the x coordinates of the head and torso are above a certain threshold, `SWAYING_HEAD_THRESHOLD` and `SWAYING_TORSO_THRESHOLD` respectively, and the z coordinates of the left and right shoulder are both below the `SWAYING_SHOULDER_THRESHOLD`.

   (c) **Turning Back Detector**: We expect there to be high variance either in the z direction of the left shoulder or the right shoulder. We also expect the distance between the left and right shoulder in the z direction to be greater when the user is turned to the side. SpeechBuddy will recognize turning back if the variance of either the left or right shoulder is above a certain threshold `TURNING_SHOULDER_THRESHOLD` and the distance in the z direction is above a threshold `TURNING_DISTANCE_THRESHOLD`.

   (d) **Hands on Hips Detector**: If a user has a hand on the hip during the presentation, we expect a small distance between that hand and the hip on the same side, below the `HANDHIP_DISTANCE_THRESHOLD`, as well as little variance in movement in the hand, below the `HANDHIP_THRESHOLD`. Therefore, we define hands on hips when we meet both requirements for either side.

(e) **Gesturing Detector**: We expect the user to be gesturing when both hands are moving, or when the variance in all directions is above a `HAND_MOVEMENT_THRESHOLD`.

(f) **Arms Crossed Detector**: Key characteristics of an arms crossed position is close proximity of opposite hands and elbows, as well as the swapping of the x directionality of the left and right hand. Therefore, we detect an arms crossed position if either distance between both pairs of opposite hands and elbows are below the `HAND_ELBOW_THRESHOLD` or the average distance between the right hand and the left hand falls below the `HANDS_CROSS_THRESHOLD`, indicating that the right hand is further left than the left hand.

(g) **Rocking Detector**: Rocking back and forth indicates a high variance in the z direction of the torso, above the `ROCKING_TORSO_THRESHOLD`. Rocking back and forth is similar to turning back to face slides, as sometimes a user may rock slightly to turn. Therefore, we also enforce a maximum distance a user can turn by requiring that the z distance between shoulders is below the `TURNING_DISTANCE_THRESHOLD`.

(h) **Hands on Face Detector**: If either hand and face of a user is below the `HAND_FACE_THRESHOLD`, then SpeechBuddy detects a hand on the face.

(i) When detector returns `true`, the feature is saved to a dictionary object called `report` with the value `true`. After analyzing the `snapshot` for all errors, the `report` object will contain only keys for the things it detected.

(j) The `report` from a single `snapshot` is then added to a `errors` object, by appending the first timestamp in the `snapshot` to the corresponding feature list.

5. After iterating through all snapshots, the `errors` object will contain mappings from each body language feature to a list of timestamps in which they occur. An example is included below.

   `errors = {"swaying":` $[time_a]$`,`
   `"turnback":` $[time_b, time_{b+1}, time_{b+2}, \ldots]$`,`
   `"gesturing": []`,
   `"handsonhips":` $[time_c, time_d, \ldots]$`,`
   `"armscrossed":` $[time_e, \ldots]$`,`
   `"legscrossed": []`,
   `"rocking": []`,
   `"handsonface":` $[time_f]$
   `}`

6. Consecutive times are grouped together into time ranges such that the first time becomes the start time and the last time becomes the end time. For each time range, a list of `formatted_error` objects are created to represent the errors.

   `formatted_error = {"start": {{START}}, "end": {{END}}}`

7. An HTML string is generated from the list of formatted errors. For each body language feature, a HTML table is generated. For each formatted error corresponding to the feature found, a row is appended to the table with the name of the feature, the start

Body Language Feature	Kinect Joint Positions
Swaying	head.x torso.x leftshoulder.z righthoulder.z
Turn Back to Face Slides	leftshoulder.z righthoulder.z
Hands on Hips	righthand.x, righthand.y, righthand.z righthip.x, righthip.y, righthip.z lefthand.x, lefthand.y, lefthand.z lefthip.x, lefthip.y, lefthip.z
Gesturing	righthand.x, righthand.y, righthand.z lefthand.x, lefthand.y, lefthand.z
Arms Crossed	righthand.x, righthand.y, righthand.z lefthand.x, lefthand.y, lefthand.z rightelbow.x, rightelbow.y, rightelbow.z leftelbow.x, leftelbow.y, leftelbow.z
Rocking Back and Forth	torso.z leftshoulder.z righthoulder.z
Hands on Face	head.x, head.y, head.z righthand.x, righthand.y, righthand.z lefthand.x, lefthand.y, lefthand.z

Table 2: Kinect Joint Positions Used to Map Body Language Features.

time of the action, and the end time of the action. If the corresponding formatted errors list is empty, then the string "None detected" is appended as a row to the table. The mp4 video created by the data formatter is also embedded into the HTML page. Javascript code is added such that when the user clicks on a row with times, the video will skip to the start time in the row.

4 Evaluation

4.1 Performance

4.1.1 Study Design

The following hypothesis was tested in the performance portion of the user study:

SpeechBuddy can detect body language errors made during presentations faster and just as accurately as a human can through conventional methods.

Feature	Threshold Name	Value
Swaying	SWAYING_HEAD_THRESHOLD	> 2000
	SWAYING_TORSO_THRESHOLD	> 2000
	SWAYING_SHOULDER_THRESHOLD	< 1000
Turn Back to Face Slides	TURNING_SHOULDER_THRESHOLD	> 2000
	TURNING_DISTANCE_THRESHOLD	> 50
Hands on Hips	HANDHIP_DISTANCE_THRESHOLD	< 150
	HANDHIP_THRESHOLD	< 50
Gesturing	HAND_MOVEMENT_THRESHOLD	> 1000
Arms Crossed	HAND_ELBOW_THRESHOLD	< 250
	HANDS_CROSS_THRESHOLD	> 0
Rocking Back and Forth	ROCKING_TORSO_THRESHOLD	> 2000
	TURNING_DISTANCE_THRESHOLD	< 50
Hands on Face	HAND_FACE_THRESHOLD	< 300

Table 3: Threshold parameters and values

The independent variable was the entity evaluating the presentation. The control condition was a human, and the test condition was the SpeechBuddy system. Humans had varying experience with giving and judging presentations, ranging from beginners to experts. The same video segment of a presentation was given to both systems as a control. The main measure of performance was accuracy in detecting and recording start and end times of the same body language features that are within the scope of SpeechBuddy, and also the quality of the free-form written feedback.

4.1.2 Study Procedure

Subjects were first given a brief questionnaire assessing their familiarity with presentation giving and judging. Then, they were given a user briefing, which detailed the purpose of the study, as well as a short training reading on aspects of favorable and unfavorable body language during presentations. Then, subjects were instructed to watch a 1 minute video of a presentation, and were asked to write down all instances and start and end times of any unfavorable body language features that the presenter exhibited. They were not allowed to pause the video or rematch the video. In the post-questionnaire, subjects were asked to submit these times, as well as free-form feedback to the presenter on strengths and weaknesses [7].

4.1.3 Results

Data from Table 4 and Table 5 seem to suggest that humans may be better at detecting the presence of an unfavorable body language feature than SpeechBuddy. 67% of subjects correctly detected all 7 features, and the other 2 subjects both missed one feature each. Both humans and SpeechBuddy had difficulty accurately recording start and stop times of body movements, with an accuracy of within 3 seconds, especially in detecting gestures.

Feature	**Accurate Detection**	**Accurate Times**
Swaying	1	0.67
Turn Back to Face Slides	1	1
Hands on Hips	1	1
Gesturing	0.83	0
Arms Crossed	1	1
Rocking Back and Forth	1	0.33
Hands on Face	0.83	0.33

Table 4: Accuracy for human subjects. Fractions are proportion of the total 6 subjects. Accurate detection refers to whether or not the subject was able to accurately determine the presence or absence of a particular movement. Accurate times refers to whether or not the subject was able to write the current start and end times within 3 seconds to all instances.

Feature	**Accurate Detection**	**Accurate Times**
Swaying	yes	yes
Turn Back to Face Slides	yes	yes
Hands on Hips	no	no
Gesturing	yes	no
Arms Crossed	yes	yes
Rocking Back and Forth	yes	yes
Hands on Face	no	no

Table 5: Accuracy for SpeechBuddy. Accurate detection refers to whether or not SpeechBuddy was able to accurately determine the presence or absence of a particular movement. Accurate times refers to whether or not SpeechBuddy was able to accurately determine times within 3 seconds.

4.1.4 Discussion

The results seem to suggest that humans were able to detect one-time body movements very well, such as placing hands on hips, and also note the time that it occurred. However, humans were often unable to pinpoint exact times for continuous motions, such as gesturing and rocking, though they were usually aware that these actions were occurring. One advantage that a human has over SpeechBuddy that should not be overlooked is the ability to analyze the talk as a whole, and synthesize these observations into a general piece of advice for improvement.

SpeechBuddy, on the other hand, performed better in detecting continuous movements over longer periods of time, like swaying, in both detection and finding start and stop times. However, SpeechBuddy is often not able to detect one time, quick motions such as bringing hands to the face with the same accuracy.

A potential concern threatening validity of this data is that the times that the accuracy of time stamps was compared against was determined by the investigator. However, the investigator has been judging presentations about once a week for 9 months, and has undergone extensive training on what to look for with respect to presentation body language. In addition, the investigator was allowed to re-watch and pause the videos to get accurate time data.

4.2 Usability

4.2.1 Study Procedure

The study was designed to assess the general usability of the SpeechBuddy system as well as the general reaction of the users to the system. Users first took a pre-questionnaire which gathered data on their past experience with giving and judging presentations. They were then given a short tutorial of how to use SpeechBuddy, and an overview of the final report UI. After the walkthrough, users were asked to give a 1 minute impromptu presentation about what they did the previous summer while using SpeechBuddy, and then use the final report to identify a few areas of strengths and improvement. After their presentation and review of the report, subjects were asked to submit a post-questionnaire on the usability of SpeechBuddy. Three subjects were recruited to test the system, all of whom were former 6.UAT students.

4.2.2 Results

Average Likert scale responses, displayed in Table 6, suggest that subjects that the tool was more useful than accurate. In general, each of the 3 subjects agreed that SpeechBuddy was useful, but the degree of accuracy differed depending on the body language exhibited by each subject. More qualitative comments submitted to the post-questionnaire suggested that the final feedback report was easy to learn, but was not immediately intuitive to navigate. All

Statement	Avg Rating
I found this tool useful.	4.67
I found this tool easy to use.	3.67
I found this tool accurate.	3.67
I would use this tool to practice for a presentation.	4.33
The feedback I received is feedback I would want to hear.	4.33

Table 6: 5-point Likert scale ratings. 1 = Strong Disagree, 2 = Disagree, 3 = Neutral, 4 = Agree, 5 = Strongly Disagree

users were confused by the time stamps, either because they were unsure what the times represented, or because the time stamps were inaccurate.

4.2.3 Discussion

Like our results from the performance study, continuous body movements exhibited by subjects were more likely to be detected and recorded than one-time, slow movements. Subjects who exhibited more continuous body movements rated SpeechBuddy more highly and found SpeechBuddy more accurate than subjects that made more one-time movements. Because SpeechBuddy relies on hard-coded thresholds determined through empirical testing, the accuracy may have also depended on how closely the body shape of the subject resembled the body shape of the investigator.

More extensive user studies with a larger sample size would need to be carried out to infirm these conclusions. Overall, users reacted positively to the system, and were pleasantly surprised.

5 Future Work

5.1 Improving Accuracy

Currently, SpeechBuddy uses time-based averaging of snapshots of video to infer how the user is behaving. However, because detection relied so heavily on changes across time, many stationary positions were left undetected. Adding a component that detects static positions may increase accuracy, but may require normalization of images or calibration to the user, like adjusting thresholds or allowing a small margin. Switching to a system that combines a static position recognition module with the current temporal based processing module into a confidence score (rather than a binary output) may lead to a more effective system.

In addition, more redundancy and specificity could be added to the body language definitions. Right now, the detections rely on a one to three joint positions, and most of the Kinect data

is unused. Finding ways to use this extra data in creative ways is the next step.

To improve accuracy for time stamps, it would be worth experimenting with different methods of extracting start and end times. Currently, SpeechBuddy looks at the first element of a given snapshot with a detected feature. The average or median time, or some combination, may also be viable options.

5.2 Expanding Feature Set

Presentations consist of many components outside of just body language. Visuals, content, speaking style, and eye contact are all extremely important aspects of a good presentation. Incorporating other dimensions, like eye contact and audio in particular would be useful to users. Audio in particular could detect fillers and an upward intonation at the end of sentences. In addition, if audio and speech were integrated, some work with transitioning between stages using speech command could be explored, in order to improve the usability of the current system of moving back and forth from the laptop.

Currently, SpeechBuddy supports 7 body language features. Future work may include adding fidgeting, leg twitching, and different sizes of gestures.

5.3 Better Feedback

A huge advantage humans have over a piece of technology is their ability to give feedback in a summarized, friendly yet constructive tone. A simple, high level summary of performance that synthesizes the data presented would be more useful than its current state.

In addition, right now all feedback is generated offline. One can imagine an extension where SpeechBuddy gives real-time feedback and makes suggestions for body language corrections while the presenter is still presenting. On the other hand, this may also be distracting, but usability tests can be conducted to confirm.

5.4 Web-Based System

Ultimately, this tool was created to make practicing presentations more easily accessible for all students. Unfortunately, a Kinect is not the easiest device to obtain. A future extension is to create a web version to do the same task.

6 Conclusion

6.1 Lessons Learned

1. There is power in redundancy and stating the obvious.
2. Be specific. Tiny nuances matter. Computers are good at following directions, not reading in between the lines.
3. Tackle a problem from multiple angles, and combine these angles into a new interpretation. The combined version is often stronger than either of the originals.
4. When user testing, the investigator will have different definitions in mind than the user will necessarily understand. User test user tests to catch these mismatches.
5. Use different technologies to take advantage of each of their strengths, instead of using one tool that will give mediocre performance in everything.

6.2 Contributions

This paper motivated and detailed the design, implementation, and evaluation of SpeechBuddy, a presentation practice and feedback system. This paper described in detail how to interact with the system. This paper also proposed a method to map raw input data from the Kinect to possible presentation body movements, and its implementation. Through user testing, this paper found that SpeechBuddy was more accurate in detecting and giving times for continuous movements, such as swaying, over one-time, slow movements, such as touching the face. This accuracy affected usability ratings. Lastly, this paper paved the way for the future by suggesting a few key extensions that would make SpeechBuddy more accurate, useful, helpful, and accessible.

References

[1] "America's Top Fears: Public Speaking, Heights and Bugs." Washington Post. The Washington Post, n.d.

[2] Cohen, Steven D. "How to Recognize & Improve Your Default Public Speaking Settings." *Harvard Public Speaking Tips: Improve Your Default Settings.* Harvard Division of Continuing Education, n.d.

[3] 6.UAT Course Info. `https://courses.csail.mit.edu/6.UAT/info.php`

[4] Hoque, Mohammed Ehsan, et al. "Mach: My automated conversation coach." *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing.* ACM, 2013.

[5] Piper, Anne Marie, Nadir Weibel, and James D. Hollan. "Introducing multimodal paper-digital interfaces for speech-language therapy." *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility.* ACM, 2010.

[6] SpeechBuddy Github Repo. `https://github.com/ericadu/SpeechBuddy`

[7] User Briefing, Pre-Q, and Post-Q. `https://goo.gl/q2gaWW`