DEVELOPING A SPEECH-BASED INTERFACE FOR FIELD DATA COLLECTION

A Thesis

by

KAITLYN CHERI BECKER

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2016

Major Subject: Visualization

ABSTRACT

This work explores the use of speech as an interface for efficient field data collection. This exploration was conducted using the open source data collection application *Field Book* as a platform. The augmented interface was created using *PocketSphinx* speech recognition and Android text to speech to enable hands-free operation of a small scope of commands. At the completion of this work, the interface concept holds promise, but has some practical limitations that need to be addressed prior to effective use.

ACKNOWLEDGMENTS

Thank you to Dr. Frederic Parke, my chair, for consistently helping me minimize the words I took to get my point across. Thank you for your patience and for helping me through one crisis after another. Thank you also to Sherman Finch, who always shows a new way to look at things, and thank you to Dr. Bruce Gooch for your expertise, guidance and friendly chats.

I wish to extend my heartfelt thanks to my parents, who have always been both my loudest cheerleaders and my most dogged motivators. Thank you for being my positive influence, my rock-solid support and the angel on my shoulder telling me to get to work.

I am so grateful for the friends and teachers I have met at Texas A&M. The lessons you taught me have been invaluable, and I'm so blessed to have met you all. Thank you to the VIZ department for putting up with me all these years, and thank you for the experience.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Many researchers today still rely on traditional log books for data collection. In scientific fields like chemistry, archeology, entomology and anthropology, the tried and true pen and paper method of note-taking is common. Accuracy and neatness are emphasized because the research is only as accurate as the data collected. As mobile technology becomes more widely available, developers are offering more efficient alternatives to traditional data collection methods. On a computer, keeping notes organized is simple, and with a mobile computing device, this technology is as portable as a paper notebook.

Field Book is an application designed by the Poland Lab as a digital note-taking tool for wheat researchers collecting data on their crops. It has been developed as part of the “One Handheld Per Breeder Initiative” which seeks to provide researchers and wheat breeders with technology that is more efficient and accurate than traditional log books (One Handheld Per Breeder). Part of this initiative is to make this technology as accessible as possible by keeping the tool inexpensive and intuitive.

In keeping with this goal, *Field Book* was designed to work with Android tablets. These relatively inexpensive and intuitive platforms have an array of useful features including Bluetooth, cameras and microphones. The *Field Book* software is open source to allow users to customize the tool to their specific needs. *Field Book* provides an interface that keeps data organized automatically and shows the data of a single plot at a time.

While *Field Book* allows faster and easier data collection, it still requires manual input. Researchers have to juggle manipulating plants and recording data, slowing down the process and often necessitating an extra set of hands. In this thesis, I have implemented an auxiliary speech interface to allow hands-free field data collection without the need for an assistant. This could save time and make the process easier and more efficient. I have created a prototype system that executes spoken commands. It also reads data aloud to verify accuracy in place of visual confirmation. This prototype system has been incorporated into the existing *Field Book* application.

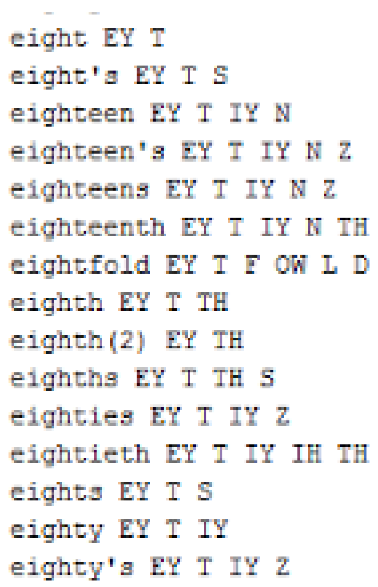
Speech interfaces have been successfully incorporated into video games, office applications, art pieces and vehicle consoles. In these diverse settings, speech interfaces are beneficial for different reasons. For vehicle consoles, having hands-free operation that doesn't take the driver's eyes off the road improves safety. In office applications, the most commonly used feature is dictation. The computer can transcribe the user's thoughts as they speak them, thus allowing users who think faster than they type to capture their message more quickly and efficiently. The main benefit of a speech interface for a video game is a wider command base. On console games in particular, there are sometimes not enough buttons to encompass the commands. As such, most games that utilize a diverse number of commands (for example, *World of Warcraft*) have to use the computer keyboard. The addition of a speech interface allows for the use of far more commands without needing extra buttons or button combinations. Finally, speech interfaces in art can be used to make a piece more interactive, drawing people in. It is a relatively new concept that hasn't been explored extensively as a media, but its use appeals to our impulse to connect to other human-like entities. If something will speak back, we probably will speak to it.

This augmentation is tailored to *Field Book,* but the information gleaned from its implementation as a multimodal interface can be incorporated in other applications. This exploration of speech-based systems can inform potential developers, artists and users of the benefits and likely drawbacks. *Field Book's* open source software serves as a convenient exemplar platform to study a system that is directly applicable to other media.

1.1 What is Speech Recognition?

The basic process of speech recognition involves three main steps. The first is to sample an incoming analog waveform to a digital representation. Next, this digital data is divided into distinct units of sound called phonemes and pauses. Finally, the resulting phonemes are run through an algorithm to determine the resulting text. The algorithm used differs between various speech recognition softwares and can have varying levels of complexity depending upon the needs of the system.

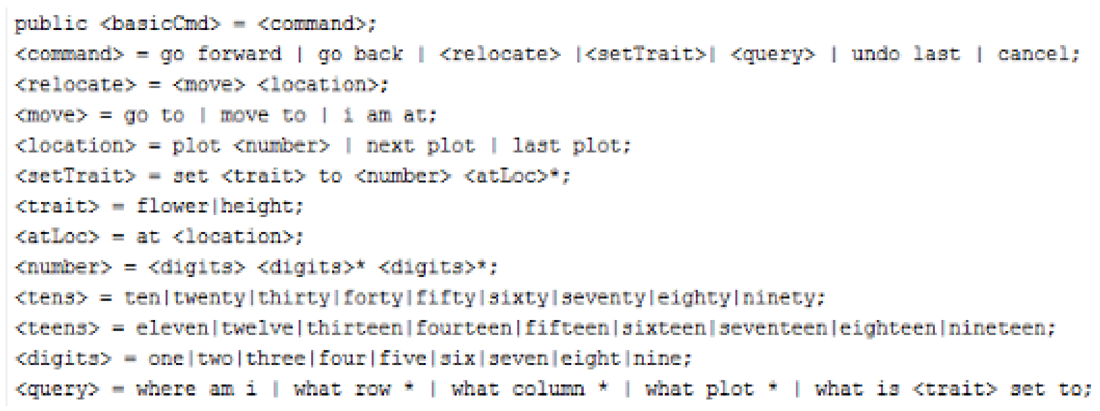
All speech recognizers have a dictionary. A dictionary consists of a list of all the words a recognizer can distinguish alongside the combination of phonemes that make up that word. It is possible for a single word to have multiple phoneme combinations just as it is possible for words in a language dictionary to have several definitions. Figure 1 below shows an excerpt from the dictionary used in this interface.



```
eight EY T
eight's EY T S
eighteen EY T IY N
eighteen's EY T IY N Z
eighteens EY T IY N Z
eighteenth EY T IY N TH
eightfold EY T F OW L D
eighth EY T TH
eighth(2) EY TH
eighths EY T TH S
eighties EY T IY Z
eightieth EY T IY IH TH
eights EY T S
eighty EY T IY
eighty's EY T IY Z
```

Figure 1. Excerpt from a speech recognition dictionary

A speech recognizer might also consult a grammar when parsing a phrase. In the context of speech recognition, a grammar is a set of guidelines to define the context in which a word can be used. For example, Figure 2 shows the grammar written for the *Field Book* augmentation. Much like a simplified version of a language grammar, it lays out the rules for when and how certain words are used. From this, the recognizer can better detect what words are being spoken by comparing them within the context of the phrase.



```
public <basicCmd> = <command>;
<command> = go forward | go back | <relocate> |<setTrait>| <query> | undo last | cancel;
<relocate> = <move> <location>;
<move> = go to | move to | i am at;
<location> = plot <number> | next plot | last plot;
<setTrait> = set <trait> to <number> <atLoc>*;
<trait> = flower|height;
<atLoc> = at <location>;
<number> = <digits> <digits>* <digits>*;
<tens> = ten|twenty|thirty|forty|fifty|sixty|seventy|eighty|ninety;
<teens> = eleven|twelve|thirteen|fourteen|fifteen|sixteen|seventeen|eighteen|nineteen;
<digits> = one|two|three|four|five|six|seven|eight|nine;
<query> = where am i | what row * | what column * | what plot * | what is <trait> set to;
```

Figure 2. Example of a speech recognition grammar

There are two main types of speech recognition: local and remote. In a local system, there is a dictionary file stored on the device that is referenced by the recognition algorithm. All computations are processed locally. These systems tend to be more accurate when dealing with a small dictionary and often have less sophisticated algorithms than remote systems.

Remote speech recognition passes the sampled waveform data to a server which will then break down the phonemes and compare them to an online repository. These systems are often more sophisticated and can recognize a large dictionary compared to local systems.

1.2 What is Speech Synthesis?

Speech synthesis, also known as text to speech, or TTS, is essentially the opposite process of speech recognition. An algorithm is used to parse text into phonemes and pauses, then phoneme sounds are produced and strung together to produce a synthetic voice. This process is outlined in Figure 3 (Andy0101). The complexity of a speech synthesis system lies in how these sounds are produced, and how accurately the digital voice mimics human speech. More complex systems will not only sound out words, but will use inflection, pauses and tonality to mimic the way we express *questions*, group thoughts in a phrase and convey emotion.

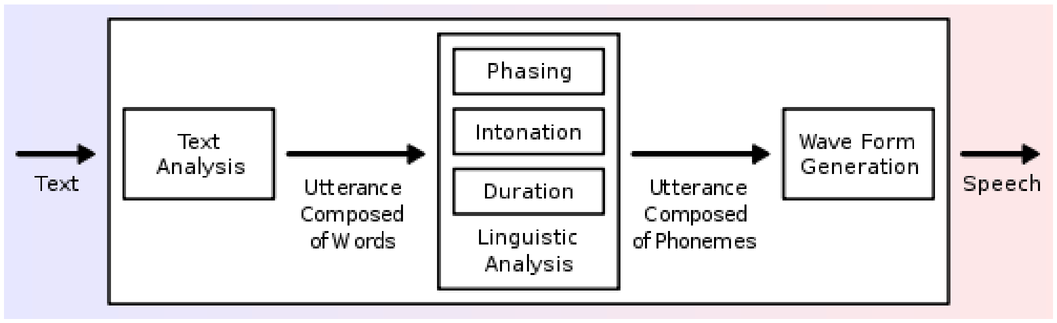


Figure 3. Illustration of a typical speech synthesis system

1.3 Problem Statement

Adding a speech interface may improve the efficiency and overall usefulness of an application. I have explored this usefulness by implementing such an interface in the *Field Book* application. I chose to focus on the benefits of making the system hands-free, having an intuitive command base, and modifying the traditional mode of interaction.

1.4 Significance

As the world's population continues to increase, food production works to keep up. Wheat is what is known as a staple food. According to the definition by the Food and Agricultural Organization of the United Nations, "a staple food is one that is eaten regularly and in such quantities as to constitute the dominant part of the diet and supply a major proportion of energy and nutrient needs" (FAO Corporate Document Repository). The FAO also states that rice, maize and wheat make up 60 percent of the world's food intake. Wheat is unrivaled in its flexibility, both as a crop and an ingredient. Wheat is grown from 67°N in Russia and Scandinavia all the way to 45°S in Argentina, from high elevations to just above sea level. It can tolerate frost, drought and poor soil conditions better than any other staple crop (P. R. Shewry). Wheat can also be stored indefinitely in the proper conditions (low humidity with protection from pests) and is easily harvested through traditional or mechanical means. (P. R. Shewry)

Researchers have developed tens of thousands of different types of wheat. New variations are constantly being produced that are more resistant to disease and pests, that are more tolerant of poor growing conditions, and that have higher germination rates and produce more grain per stalk. This ongoing research is imperative to keep up with the growing demand for food.

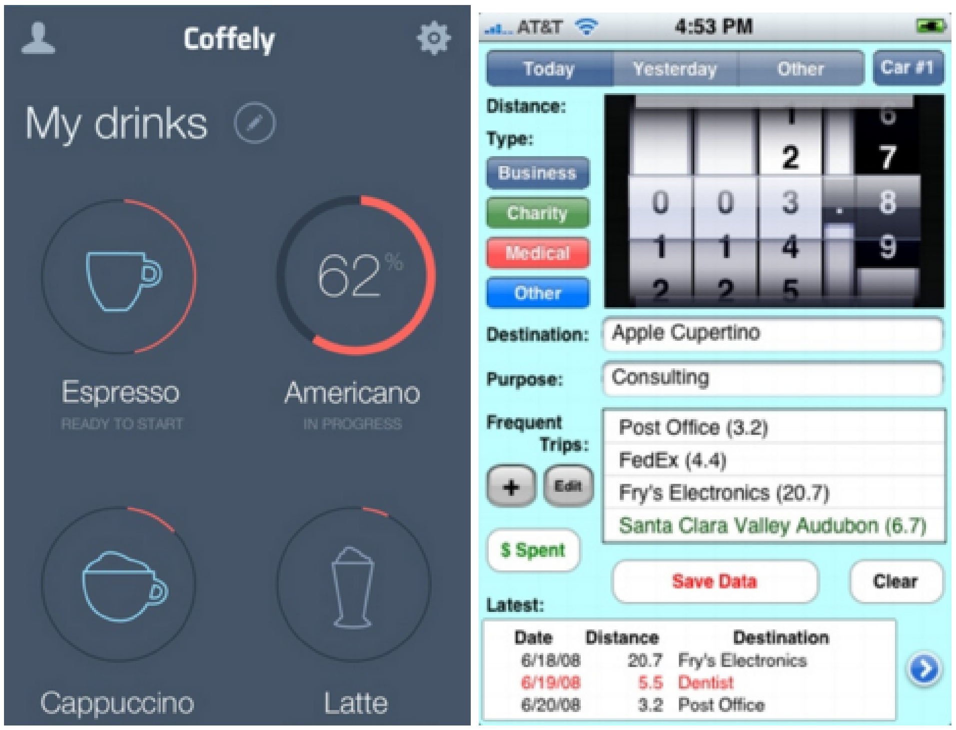
Field Book is making data collection easier for researchers. By providing another option for controlling the application, data collection may be made more versatile and efficient. With a more flexible and robust interface, the application could be more accessible, thus making *Field Book* a more approachable option for researchers.

1.5 Motivation

The overall goal of the *Field Book* application is to make data collection more efficient and accessible. The current iteration of the software meets this goal, but with the addition of a speech-based interface, many interactions may be made easier. Adding a speech-based interface would not increase the cost of the system as most mobile devices come equipped with audio input and output devices.

Currently, users operate the application with touch input, requiring both hands. Researchers have to juggle between manipulating equipment, handling plants and inputting data. According to field researchers at Texas A&M, the usual approach is to have one person dictating the data and another transcribing it (Brian Pfeiffer, personal communication, June 4, 2015). With the inclusion of a speech interface, the application may be operated hands-free, thus streamlining the process and eliminating the need for an assistant.

User interface design is all about minimizing the work needed to complete a task. A good user interface has only necessary elements on screen. A cluttered interface leads to inefficient navigation and takes longer to operate. Figure 4 (Qayyum, Stevens Creek) illustrates this point. Speech-based control systems can reduce the time and effort needed. Users can activate a response without a specific button, minimizing screen clutter.



Distance:
Type:
Destination: Apple Cupertino
Purpose: Consulting
Frequent Trips:
- Post Office (3.2)
- FedEx (4.4)
- Fry's Electronics (20.7)
- Santa Clara Valley Audubon (6.7)

Latest:

Date	Distance	Destination
6/18/08	20.7	Fry's Electronics
6/19/08	5.5	Dentist
6/20/08	3.2	Post Office

Figure 4. A simple interface (left) is easier to navigate than a cluttered one (right).

Speech-based control may also reduce the time needed to navigate menus and submenus. For example, to access a non-adjacent plot using the current version of *Field Book*, the user must expand a drop-down menu, open the map, wait for it to initialize, and then count through the unlabeled cells to find the plot they're looking for. With the implementation of speech-based commands, the user can just say 'Go to plot 285'.

Field Book currently allows users to record speech for later transcription. This feature is not commonly used by researchers because of the time it takes to later transcribe these notes (Trevor Rife, personal communication, May 19, 2015). With speech recognition, the user would have a written record that could be transmitted to colleagues immediately without the transcription time.

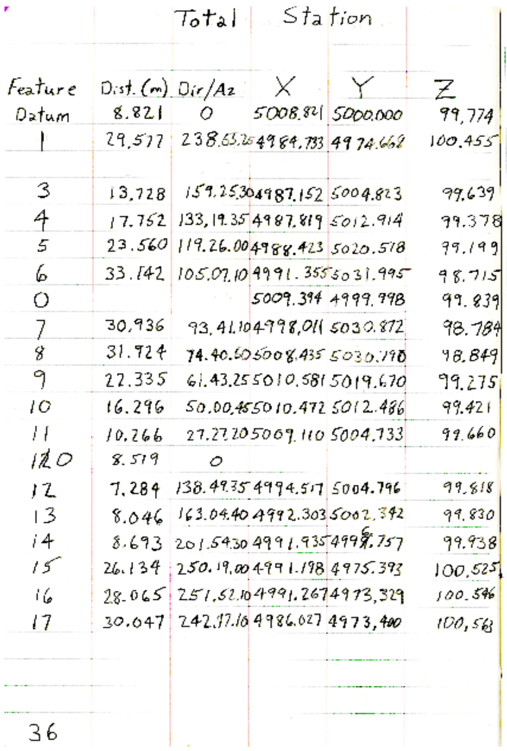
By utilizing speech-based software developed for multiple languages, the interface could also be made available to non-English speaking researchers. With the addition of speech synthesis, the app could read aloud (in the native language) instructions or descriptions to the user, making this app accessible to those with limited literacy. Spoken text could also be added to the in-built tutorial. Since the system is most often used in bright sunlight (Trevor Rife, personal communication, May 19, 2015), read aloud functionality could be useful when screen readability is poor.

2. PREVIOUS AND RELATED WORK

2.1 Data Collection

Since the advent of modern agriculture, field researchers have used paper to record their data. Survey books with waterproof pages made to fit in a pocket are often used. There are conventions as to how these pages are laid out and how the data is recorded.

Figure 5 shows an example of data layout (Doolittle). Traditionally, data in columns are on the left, and figures and sketches are on the right. The necessity for precision makes recording in a survey book a time-consuming endeavor. Researchers have the option of designing and reproducing a form that is custom tailored to their specific project. This requires the researcher to know ahead of time what data they will be collecting and offers little flexibility.



Total Station

Feature	Dist. (m)	Dir/Az	X	Y	Z
Datum	8.821	0	5008.821	5000.000	99,774
1	29,577	238.65.25	4984.733	4974.662	100.455
3	13,728	159.25.30	4987.152	5004.823	99.639
4	17.752	133,19.35	4987.819	5012.914	99.378
5	23.560	119.26.00	4988.423	5020.578	99.199
6	33.142	105.07.10	4991.355	5031.995	98.715
0			5009.394	4999.998	99.839
7	30,936	93.41.10	4998,011	5030.872	98.784
8	31.924	74.40.50	5008.435	5030.798	98.849
9	22.335	61.43.25	5010.581	5019.670	99.275
10	16.296	50.00.45	5010.472	5012.486	99.421
11	10.266	27.27.20	5009.110	5004.733	99.660
1~~2~~0	8.519	0			
12	7.284	138.49.35	4994.517	5004.796	99.818
13	8.046	163.04.40	4992.303	5002.342	99.830
14	8.693	201.54.30	4991.935	499~~9~~6.757	99.938
15	26.134	250.19.00	4991.198	4975.393	100.525
16	28.065	251.52.10	4991.267	4973,329	100.546
17	30.047	242.17.10	4986.027	4973,400	100,563

36

Demonstration / Activity

Geography Parking Lot
13 October 2000
Cool, calm, PC
W. Doolittle P.C.
373F Class

Figure 5. An example of the data organization system commonly used in logbooks.

2.1.1 Mobile Applications for Data Collection

Users are becoming more comfortable using technology for tasks previously done on paper. According to a survey conducted by Princeton Survey Research Associates International, 50% of American adults own and use either a tablet or e-reading device (Zickuhr and Rainie). In recent years, researchers have begun adopting new technology in an attempt to make the process of data collection easier and more streamlined. With advances in mobile technology, digital tablets are becoming more useful as data recording media. They offer the portability of a logbook with the computational power of a computer. With a well-written application, a user can edit, reorganize and customize their recorded data, as well as disseminate the gathered information efficiently.

Of the researchers who use a tablet as part of their data collection, many rely on Microsoft Excel. Excel offers a customizable grid and powerful mathematical functionality. Averages, totals and other survey data can be calculated and updated on the fly and can be translated into graphical format to help visualize information.

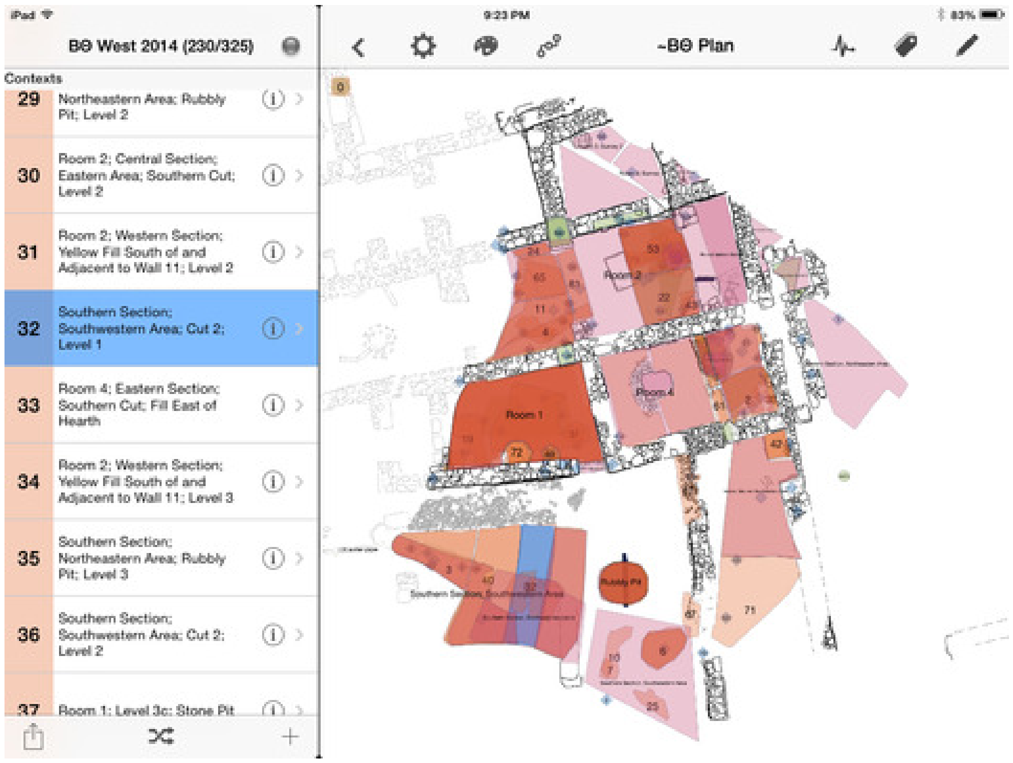
Excel is useful for data visualization and organization, but when it comes to mobile data collection, it proves unwieldy. Data cells and input keys are small and can be difficult to press accurately. The display is nearly impossible to see in bright lighting conditions on tablets with inadequate glare reduction. The grid format for information is inefficient on large plots due to the standard serpentine order of collecting. It can also be difficult to maintain the correct position in a spreadsheet when it is necessary to skip cells that do not have data to be input.

Many applications have been developed in the past decade to mitigate these limitations. Some focus on global data management, allowing collaborators to pool their collective knowledge. Applications like Magpi (DataDyne Group) and Epicollect (EpiCollect.net) provide tools for creating mobile data collection forms and include functionality like GPS location and photo uploads. While they support data collection, their primary goal is organized dissemination of data.

Reference guides are particularly well-suited to mobile application development. Some, like Plant-o-matic (Ocotea Technologies, LLC) and the iBird Guide (Mitch Waite Group), present the user with a powerful search tool with which they can identify a specimen. Other references, like Project Noah (Networked Organisms), rely on the contributions of 'citizen scientists' to report and identify sightings of fauna across the globe. The application serves as a hub for a community of users to share their findings or seek help from their fellow researchers. There are also automated references with photo-processing technology. Applications like LeafSnap (Columbia University, University of Maryland, and Smithsonian Institution) make use of image recognition technology to analyze a photo of a specimen and return an identification.

2.1.2 Mobile Applications for Specific Fields

While many applications exist that allow users to create their own surveys and forms, applications are also developed for specific types of data collection. These applications are optimized for the type of information to be collected. Considerations are made for the specialized environment in which the tool will be used. For example, the application iDig (Bruce Hartzler) is a tool developed for archeologists. Users can easily scan sketches and maps, upload custom data points, and can wirelessly sync with other iPads on the site. Plans can be viewed in cross-section and from above, and the application allows for geospatial searching of archival excavation data. A screenshot of this application is shown below in Figure 6 (Hartzler).



BΘ West 2014 (230/325)

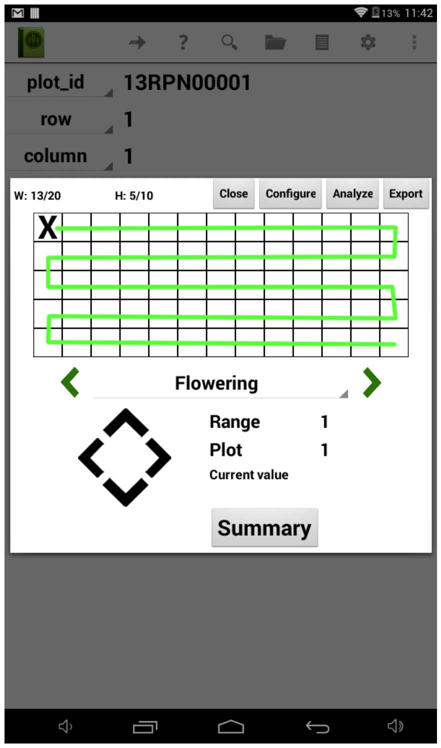
Contexts

29	Northeastern Area; Rubbly Pit; Level 2
30	Room 2; Central Section; Eastern Area; Southern Cut; Level 2
31	Room 2; Western Section; Yellow Fill South of and Adjacent to Wall 11; Level 2
32	Southern Section; Southwestern Area; Cut 2; Level 1
33	Room 4; Eastern Section; Southern Cut; Fill East of Hearth
34	Room 2; Western Section; Yellow Fill South of and Adjacent to Wall 11; Level 3
35	Southern Section; Northeastern Area; Rubbly Pit; Level 3
36	Southern Section; Southwestern Area; Cut 2; Level 2
37	Room 1; Level 3c; Stone Pit

Figure 6. The iDig application was developed specifically for archeologists.

2.1.3 Field Book

Field Book is engineered specifically for field data collection. The application was developed with the needs of wheat researchers in mind. Users create a grid in Microsoft Excel, or other spreadsheet software, detailing the title or id (usually a number), row and column of each plot and import it to the tablet. *Field Book* then generates a map of the field. When creating a new data set, users specify the data points, referred to as 'traits', they will be collecting (e.g. flowering date, height, disease rating), and the name of the researcher inputting the data. *Field Book* then allows the user to input the data per plot. Traditionally, researchers follow a serpentine pattern when collecting data; completing one row in ascending order, then following the next row in descending order. An example of this path is shown in Figure 7. *Field Book* assumes this layout when progressing through a data set. *Field Book*'s visual design is high contrast with large text and large buttons to facilitate its intended use in bright sunlight. Inputting data is streamlined, making data collection faster and easier than traditional paper methods. Users can export their collected data in spreadsheet format, allowing them to make use of Excel's mathematical capabilities.



plot_id 13RPN00001
row 1
column 1
W: 13/20 H: 5/10
Range 1
Plot 1

Figure 7. *Field Book* data input follows a serpentine path through a field.

Field Book maintains flexibility for users by allowing them to define the traits they wish to record for a specific crop. The user can create data points of many different types including numeric, date, text, photo and audio. This allows the user to fine-tune the process to suit the needs of different research projects.

2.2 Speech-Based Interfaces

2.2.1 A Brief History of Speech Systems: Speech Recognition

The first documented speech recognition program was Audrey (K. Davis). Audrey was built in 1952 at Bell Laboratories. This computing system could recognize ten numbers, “0” through “9” and only after the system had been trained to the user’s voice and with significant pauses in between. Audrey worked by comparing each sound to a library of individual sounds to find a match. At the time, the system was physically too large, too slow and too expensive to compete with faster and cheaper push-button interfaces. Ten years later, at the 1962 Seattle World's Fair, IBM released the “shoebox” computer (IBM archives). This system still required awkward pauses in between words, but it was able to recognize 16 words in addition to single digits.

In the 1970s, the United States’ Department of Defense began its Defense Advanced Research Projects Agency (DARPA) Speech Understanding Research (SUR) Program. The most successful project funded by the SUR program was Harpy, developed at Carnegie Mellon University (Bruce Lowerre). Harpy used a new searching algorithm called “Beam search” that enabled it to recognize over 1000 words, roughly the average vocabulary of a three-year-old. Soon after, in the 1980s, researchers began investigating a recursive technique known as the Hidden Markov Model. This model was first described by Ruslan L. Stratonovich in 1960 (Stratonovich), and gained popularity when researchers at Bell Labs began implementing it. The Hidden Markov Model allowed computers to assign probabilities of individual sounds being words rather than comparing them against a repository. This flexibility paved the way for much larger vocabularies.

With advances in personal computing technology, commercial software was soon developed. In 1997, Dragon introduced their *Naturally Speaking* software, which allowed users to speak at a conversational pace rather than the stilted speech that was previously necessary (Nuance Dragon Naturally Speaking). DARPA launched the Effective, Affordable, Reusable Speech-to-Text (EARS) project in 2002. The goal of this project is to improve the accuracy of multilingual speech-to-text transcriptions from phone conversations (Effective, Affordable, Reusable Speech-to-Text). Soon after, in 2005, DARPA began work on the Global Autonomous Language Exploitation (GALE) project. The goal of GALE is to develop a system that can automatically process recorded conversation, text documents and news broadcasts in a variety of languages and organize the data in a queryable format (Global Autonomous Language Exploitation). In 2007, Google began its research in speech recognition with the GOOG-411 system, a telephone-based directory service (GOOG-411 Team). This eventually led to the connection of speech recognition to cloud-based technology, massively increasing the capabilities of modern systems. With the massive computational power and vast data stores available today, accuracy in current systems are frequently above 90% (Jordan Novet).

2.2.2 A Brief History of Speech Systems: Speech Synthesis

Speech synthesis has been around far longer than recognition systems. The first known efforts to produce a synthetic voice were made in 1779 by Professor Christian Kratzenstein (EduBilla). Professor Kratzenstein studied the human vocal tract and made artificial resonators that could produce five vowel sounds. Around the same time, Wolfgang von Kempelen was working on his own speech machine. In 1791, he introduced his “Acoustic-Mechanical Speech Machine” which could produce single sounds and a few sound combinations (Wolfgang Kempelen). The machine was constructed with parts analogous to the working of the human body, including lungs, vocal cords, vocal tract, tongue and lips. In the 1800’s, Charles Wheatstone unveiled a more complicated version of von Kempelen’s machine (James Flanagan). Wheatstone’s machine was capable of producing most vowel and consonant sounds and was even able to produce full words.

In 1922, Stewart released the first fully electrical synthesis device (Haskins Laboratories). The system used electricity to create resonance and was only able to make vowel sounds. The first electrical device that could produce intelligible synthetic speech was introduced at the 1939 New York World’s Fair by Homer Dudley (Homer Dudley). Dudley was inspired by the VOCODER, a device built by Bell Laboratories in the 1930’s that could analyze input speech into parameters that could be used by a synthesizer to reconstruct the original sound. Dudley called his synthesizer VODER. The system was complex and difficult to play, and the resulting speech was only barely intelligible.

At Haskins Laboratories in 1951, Franklin Cooper and his team developed a Pattern Playback synthesizer that reconverted recorded spectrograms into sound (Haskins Laboratories). George Rosen filed a patent for the first articulatory synthesizer in 1958 (George Rosen). This synthesizer was controlled by a tape recording of signals produced by hand. During the 1960s, research was conducted on Linear Predictive coding, in which the formants (vowels) of an input sound are separated from any sibilant (hissing) and plosive (popping) sounds in the recording. By running the source through a filter created from the formants, speech is produced. This model was used in early low-cost systems like Texas Instruments' Speak'n'Spell from the 1980s. This predictive system is still used today.

The first text to speech program in English was developed in Japan in 1968 by Noriko Umeda and his associates (Umeda N.). It used the articulatory model with an added syntactic analysis model and was regarded as intelligible, though monotonous speech. A team of researchers at M.I.T. demonstrated the *MITalk* system in 1979 (Jonathan Allen et al.). This system was later modified for use as a commercial system for Telesensory Systems Inc. In 1981, Dennis Klatt introduced the famous *Klattalk* system (Dennis Klatt). These two systems formed an early basis for speech synthesis technology that is still used today.

2.2.3 Current Uses of Speech Interfaces

Speech recognition has been used as a dictation tool for many years for both casual and office use. It has proven to be an accurate and efficient tool for transcription of medical reports, reducing average turnaround time from 28 hours to 12.7 hours in a study conducted in the Department of Radiology at the University of North Carolina Hospital (Krishnaraj et al.).

Speech recognition has also been used for command interfacing. Many computers and most smartphones come with the ability to execute voice commands. This powerful hands-free capability makes it possible to multitask when driving or cooking, and improves accessibility for those with limited mobility or dexterity. Most of these systems include speech feedback to facilitate hands-free operation. Call centers use speech recognition for directing millions of calls each day.

Video games have also made use of speech recognition technology. Games like *Hey You, Pikachu* (Nintendo) or, more recently, *There Came an Echo* (Iridium Studios), use voice commands to control the interactions between the computer and the player. This frees up screen space and allows for more commands than the interface has buttons (in the case of console games). It also serves to make control more intuitive. In the case of *There Came an Echo*, the player acts as a commander of a small unit of soldiers. The commands mimic those of an officer in charge, such as “Move to point A”, “Target zone 5”, and “retreat”. This gives the game a more immersive feel than with point and click controls. It has the drawback of being slightly slower, and can be frustrating when the commands get repetitive, but it can make you feel like you’re a part of the game.

3. METHODOLOGY

To explore the usefulness of a speech-based interface for the interactive *Field Book* application, I created a prototype that extends *Field Book* to include constrained speech recognition for command execution and audio feedback.

I visited a sorghum field with Texas A&M researchers and observed the use of *Field Book* in its intended environment as my first step. On the trip, I noticed several inconveniences that could be mitigated by a speech interface. For one, the tablets require two hands to input data. This can be inconvenient for users who must also handle the plants or other equipment (like a measuring pole). The researcher also mentioned that the tablet touch screens tended to be less responsive in rain or mist, making data difficult to input. Inputting data also requires the user to look at the screen. It can be difficult to see the screen when the sun is very bright. With a speech system, input is handled via a microphone, minimizing the need to handle the device or view the screen.

To ascertain the most intuitive setup, I framed the system as a kind of digital secretary. I asked the researchers to go over how they would relay their information if they had an assistant. Having one researcher call out information and one record it is common practice for large fields, as it speeds up the process. I made note of when the speaker would announce information, and when the recorder would ask for confirmation. In the case of a speech recognition system, particularly the less accurate local variety, it was necessary to assume this digital secretary was rather hard of hearing and would have to ask for confirmation more frequently than a human counterpart.

4. IMPLEMENTATION

4.1 Design Considerations

In designing an augmentation for *Field Book*, it was important to take into consideration the current implementation of the software. The application was designed for 7 inch tablets to maximize available screen space while keeping the device small enough to be easily portable. It was first developed for Android systems because the cost of these devices is lower and distributing the application is free and easy. This is in keeping with the goal of the “One Handheld Per Breeder” initiative. By keeping costs low, the application is available to more researchers. Android offers tablet models that are constructed to be durable, making them practical for field work.

A potential challenge faced by an outdoor speech recognition application is input sound quality. In practice, there likely will be environmental noise from wind, nearby roads or other interference. Distance of the user from the microphone (for speech recognition) or the speakers (for audio feedback) can reduce the user’s ability to properly communicate with the system. The user could hold the tablet closer to their head, but this defeats the purpose of hands-free functionality. As such, I worked with an inexpensive headset to provide quality audio input and output.

It is assumed that researchers may not have internet access while recording data in remote fields. As such, the data collection capabilities of the software have no internet dependency. Data dissemination is handled outside the application following data export.

The *Field Book* application is being introduced to both new and seasoned researchers. Prospective users run the gamut from avid technical gurus to traditional pen and paper enthusiasts. To account for this, developers have worked to make the interface as intuitive as possible, including an on-board tutorial.

In keeping with these design considerations, I developed this speech augmentation for the *Field Book* application on a Neutab N7 Android tablet. My prototype requires little added equipment cost. The text and buttons added to the interface are large and high contrast to be seen easily in bright sunlight. The speech recognition and speech synthesis softwares used (*PocketSphinx* and Android Speech Synthesis, respectively) are available for free and operate locally without access to the internet. I made efforts to keep the system as intuitive as possible, relying on common-sense phrasing for commands and accounting for some variation.

4.2 Identifying Useful Commands

The primary goal of this speech augmentation is to make data collection a hands-free operation. It is important that the user not feel the need to check the accuracy of each entry. To accomplish this, audio feedback in the form of speech and tones has been implemented in addition to speech commands.

To determine a useful set of commands, I accompanied two wheat researchers from Texas A&M on a survey of their project fields. The process for recording data is illustrated in Figure 8 below.

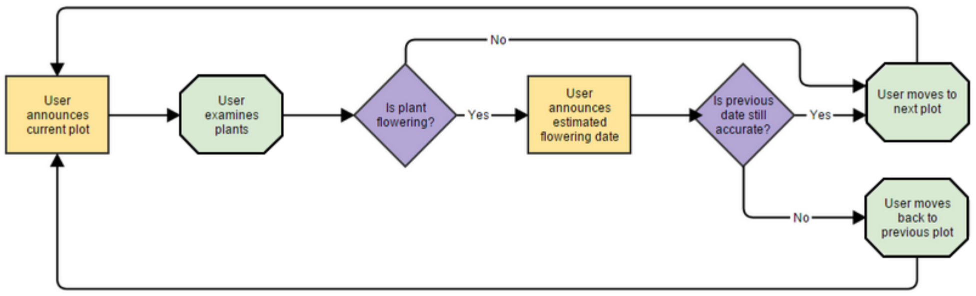

User announces current plot
User examines plants
Is plant flowering?
Yes
No
User announces estimated flowering date
Is previous date still accurate?
Yes
No
User moves to next plot
User moves back to previous plot

Figure 8. The process for collecting flowering dates of Sorghum plants in the field

The two actions shown in rectangles are points at which a researcher would announce to their assistant what to write down. These actions served as my initial reference when developing commands for the prototype.

The first allows the user to state at which plot they are recording information. The use of this command will feel natural as researchers will often count off plots as they walk them. The system would then automatically keep pace with the researcher rather than the researcher keeping track of which plot they have selected on screen. This command can be a simple “move forward”, “move backward”, “go to next plot”, or the user can specify the plot number: “go to plot 100”.

As is shown above, researchers sometimes have to overwrite previous data as more information becomes available. For example, a researcher says that a particular plot must have flowered a day ago based on the appearance of the panicle (the loose, branching cluster of flowers found at the top of the plant) and the amount of pollen it has released. The researcher then encounters a plot a few columns down that has released less pollen but clearly didn’t start today. The researcher records that the current plot flowered yesterday and edits the previous plot to say it flowered two days ago. Collecting data that requires estimation, like flowering date, is subjective and users must be able to go back and modify the data if they change their minds. The system can understand multiple phrases to accomplish this, including “Go to plot 100” or “I am at plot 100.”

The second action a user would announce is what trait to record and what its value is. To be most intuitive, this command mimics the researcher announcing data to an assistant. For the purposes of this prototype, I have implemented the most common type of trait input: numeric. Currently, the user can record height data or flower date in epoch form. The user can either navigate to a plot and then record the data (“Go to plot 120. Set height to 40[inches]”), or command inclusively (“Set height to 40[inches] at plot 120”).

The system asks for and awaits confirmation before changing data. The system announces the registered command and prompts (e.g. “Want to set height to 40 at plot 120?”), and waits for a ‘yes’ or ‘no’ answer. If the user answers no, the system will discard the changes and await further instruction.

The user can also undo the last step. If the system moves or sets a trait incorrectly, the user can simply say “Undo last”. This is necessary as we can’t assume the system will always be 100% accurate. I chose to use the phrase “undo last” rather than just “undo” to avoid false triggers.

Finally, the researcher has the ability to request that the system read out the data for a given data point, whether that be the current plot id, row, column, or traits of the current plot. Examples of working commands include: “Where am I?”, “What row is this?” or “What is height set to?”.

4.3 When Audio Feedback is Necessary

Since the user should not feel the need to look at the screen when using this interface, audio feedback becomes the primary way of communicating information. It is necessary for the user to know when the system is listening, when the system recognizes or fails to recognize a command, and when it requests confirmation. To accomplish this, the system provides audio feedback for each case. This system makes use of both speech and tonal feedback, depending on the situation.

The system is always listening, but only activates command detection when it is specifically addressed using the key phrase “Field Book”. This prevents the system from trying to interpret everything it hears as commands. When the application detects the key phrase, the system issues a notification sound to alert the user. Here, sound was employed rather than a spoken phrase to minimize the time between the user activating the system and speaking their command.

When the system recognizes a command, it issues a unique notification tone, then repeats the command it heard. For example, if the system recognizes the command ‘move forward’, plays the command recognized sound (a synthesized chime), then says “Moving forward to plot [next plot]”. In the case of any error, the system will first play an error notification sound (a descending pair of tones). If it fails to recognize a given command, it says “[command issued] is an invalid command.” If the command has a problem, for instance if the user attempts to move to a plot outside the map’s bounds, the system will speak to the specific problem, in this case “The maximum plot is [# of plots], cannot access plot [the plot the user attempted]”.

It is necessary for the researcher to receive audio feedback to ensure accuracy of data collected. For the trait command, the system announces the trait and value it is going to record, as well as the plot it is recording in. The system awaits feedback in the form of a ‘yes’ or a ‘no’ from the user prior to making changes in case the data is incorrect. The same occurs for the move command. Unlike other commands, the move forward and move backward commands do not await confirmation as they are used to move quickly between plots. The potential risk of falsely triggering these commands are combated by their dissimilarity to other command phrases. Figure 9 shows the process of the system executing a command.

The user can request data such as the current row, column, plot, and trait values of the current plot using the commands detailed in the previous section.

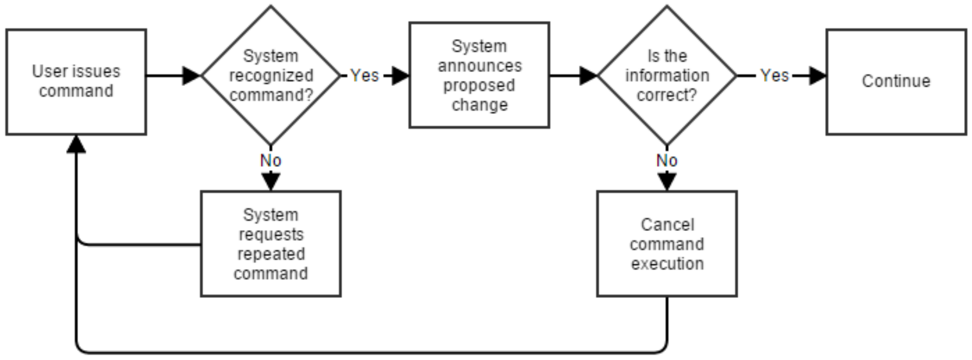

User issues command
System recognized command?
Yes
No
System announces proposed change
Is the information correct?
Yes
No
Continue
System requests repeated command
Cancel command execution

Figure 9. The system processing a single command

4.4 Hardware and Software

This software was tested on a Neutab N7 Android tablet. This tablet was inexpensive while still providing the features necessary for testing, including adequate processing power, screen resolution and sound quality. It had a headphone jack for the inexpensive headset and debugging capabilities. The headset I used was a set of Apple earbuds with built in microphone.

I built my program in Android Studio, the official IDE (Integrated Development Environment) for Android application development. I began by creating a rudimentary program to get accustomed to application styling and work through the kinks of getting the tablet to talk to the computer. With that Hello World completed, I moved on to text to speech. Text to speech is included in the base Android environment, so it was not necessary to seek third party software.

Speech recognition proved far more difficult. Like text to speech, speech recognition is available with base libraries for Android, and is, on some versions, capable of working offline. I was never able to get it to work. Instead, I moved over to *PocketSphinx*. *PocketSphinx* is an open source, local speech recognition application that allowed me to define my own grammar, thus limiting the number of words my application would try to recognize and specifying how they would be arranged. By limiting the options, the recognizer was more accurate. It also meant that I knew the exact format of recognized commands and could use them accordingly. Because it is open source, it can be incorporated into *Field Book* without licensing issues or adding cost to the system.

I found there were significant drawbacks to using the *PocketSphinx* software. One example was the lack of control over audio threshold. There is no built in way to tell *PocketSphinx* to discard low level sound information. *PocketSphinx* adjusts its recording threshold automatically and tends to amplify environment noise and get false positive results. As such, I found it necessary to change the activation key phrase from the single syllable ‘book’ to the longer ‘Field Book’ to eliminate some of those false positives. I also directed the command listeners to reset after a short period of time. This helps prevent the system from over-amplifying prolonged silence to produce errant results.

Another challenge was the inherent inaccuracy of the recognizer. At times, it would detect every word flawlessly, and at others it would fail. To combat this, I sought to make commands phonetically diverse, so no two commands were similar enough to be easily confused. I also switched from my original grammar that allowed for intuitive input of numbers such as “one hundred and one” to digits “one zero one”. This was necessary because the system couldn’t reliably distinguish close-sounding words like “sixteen” and “sixty”. I ensured the system repeated any command it was attempting and for any command that would change data to await confirmation prior to executing.

The software also tended to have unpredictable periods of lag. These would happen most often when the system would hang while trying to interpret a speech input. My attempts to optimize the system have thus far failed to alleviate this unfortunate behavior, so more time must be spent determining the cause.

5. RESULTS AND CONCLUSIONS

The conclusion I reached conducting this study is that getting a speech interface to work properly is more difficult than I had anticipated. Having done the background research and played with tools like *Siri*, I felt I was adequately cynical about the capabilities of such a system. I was not.

There are many factors that make implementing a speech interface a challenge. First is the number of points at which the system can fail. In a traditional digital system, a button is essentially a yes or a no. A command in a speech system can be several hundred yes's and no's. Because of this, a speech-based command system must have a commensurate number of fail safes. It is up to the designer of the system to anticipate these failure points and account for them. Failure rates in a speech recognition system increase as commands are less distinct (e.g. 'sixty' vs 'sixteen'). Even with widely distinct commands, the current system misinterprets speech roughly once in five attempts. This can be incredibly frustrating to deal with and would require improvement for the system to be used in the field. While there are error checks in place to prevent the system from executing an errant command, there is a significant delay in having to cancel and restate a request.

The second challenge when working with a speech-based system is finding the appropriate software (assuming the user is not making their own). When conducting that search, the user must know what is needed for their system. For this implementation, I needed a software that was open source and reasonably accurate without continuous access to the internet. I also strongly preferred any system with a customizable grammar.

After finding that software, the designer of the system must be able to work around that software's idiosyncrasies. For instance, *PocketSphinx* doesn't give the user the ability to set the minimum threshold for sound, so it will pick up noises that are not commands. I dealt with this using timeout functions to cut any noises that hadn't been identified as a command. *PocketSphinx* is well documented and has an active support forum, so finding help wasn't difficult. In other cases, however, dealing with poor documentation or lack of support could make a software virtually unusable.

The second conclusion I came to is how helpful it is to have a good mental model. In designing this interface, I found that thinking of my speech interface as dealing with a person who is hard of hearing helped me visualize how the 'conversation' should play out. I found it easier to see where potential miscommunications could occur and how to address them. Thinking about the system in terms of a conversation was a pragmatic way to consistently come up with intuitive command phrasing.

In exploring this speech-based system, I worked out a series of do's and do not's for those considering the use of a similar system:

Do expect a good speech-based control system to feel like a conversation. Because this type of system requires the user to talk to it, and for the system to talk back, it inherits many of the characteristics of a traditional conversation. It can be more engaging to users than a passive point and click system. With a large enough command base, this type of system is also intuitive. The user can request a specific action, and the system will return the expected result.

Do expect a hands-free experience. Given the appropriate command base and audio feedback, the user does not have to physically interact with the system at all except for listening and speaking. This leaves the user free to use their hands and eyes elsewhere.

Do the research for what type of recognizer your system will need. Be aware of the requirements of the proposed system. Does it need a free or open source software, or will it need a proprietary package? Know the priorities of speed, accuracy and flexibility. Will the developer define the grammar or use an open ended phonetic interpreter? A wider command base can mean less overall accuracy. Does the system require access to the internet? Internet dependent systems have access to more processing power and can handle larger dictionaries, but they can also be slower or less reliable depending on your connection.

Do expect to have to update your system. The first iteration (and likely the fifteenth) will not be perfect. Be prepared to make changes to the command base, the grammar, and the command interpreter many times to get it to a useable state.

Don’t expect the recognizer to be completely accurate. Even the best recognizers have some degree of error. Always prepare your system for that eventuality and make sure it can handle miscommunications. It is far better to prepare for failures that never happen than to not be prepared for the one that does.

Don’t expect the recognizer to be instant. Speech recognition takes time. The fastest systems still require a second or so to process. Speech controls are therefore not appropriate for situations requiring twitch controls such as in first person shooter games.

Don’t get discouraged. Speech recognition is a conversation. It doesn’t always go the way you’d expect. Be prepared to deal with the frustration of fine tuning the system to get it to a point where it is useable.

Overall, this system does what it was expected to do. It facilitates the use of the *Field Book* application as a hands-free tool. The user can move, set traits and “view” information without having to handle or look at the device. With some improvements, this system could function as a useful, not just useable, addition. It currently has the capability to execute the commands tasked to it, but it is incorrect approximately 1 in 5 attempts. This causes significant frustration for the user, who would likely dismiss it for the more efficient option of manual input. To be useful, the system would require higher accuracy, fewer false positive results, and would require reduced lag time. If these requirements were met, the system could be reliable enough for the benefits of a hands-free interface to outweigh the unfavorable traits.

6. FUTURE WORK

For future improvements of the system, I would like to continue to refine the accuracy of the recognizer. The incorporation of a remote recognizer could provide better translation while adding the requirement of continuous internet access. The system could also potentially incorporate user profiles, which would improve accuracy by training the system to a specific user's speech traits. I would also like to expand the commands the system is capable of handling, including the ability to navigate the main menu, load maps, and create new traits.

I would also like to continue to explore the cause of the unpredictable lag and remove it. Further optimization of the code structure would help with this, as well as seeking help from the developers of *PocketSphinx.*

More expansion possibilities include translation to other languages, and dictation for notes. In an ideal setup, users could add customized commands through an intuitive interface rather than having to modify the code.

REFERENCES

Allen, Jonathan, Sharon Hunnicutt, Rolf Carlson, and Bjorn Granstrom. "MITalk-79: The 1979 MIT Text-to-speech System." *Acoustical Society of America* 65.51 (1979): n. pag. *AIP Scitation*. AIP Publishing LLC, 11 Aug. 2005. Web. 29 Sept. 2015. <http://dx.doi.org/10.1121/1.1906946>.

Andy0101. *Overview of a Typical TTS System*. Digital image. *Speech Synthesis*. Wikipedia, 13 Mar. 2012. Web. 20 June 2015. <https://en.wikipedia.org/wiki/File:TTS_System.svg>.

Davis, K. H. "Automatic Recognition of Spoken Digits." *J. Acoust. Soc. Am. The Journal of the Acoustical Society of America* 24.6 (1952): 627-42. Web. <http://dx.doi.org/10.1121/1.1906946>.

Doolittle, William E. Logbook data organization. Digital image. *Recording Data*. University of Texas, 12 Dec. 2013. Web. 23 June 2015. <http://uts.cc.utexas.edu/~wd/courses/373F/pdf/4.4FullPage.pdf>.

Dudley, Homer, R.r. Riesz, and S.s.a. Watkins. "A Synthetic Speaker." *Journal of the Franklin Institute* 227.6 (1939): 739-64. Web. <http://dx.doi.org/10.1016/S0016-0032(39)90816-1>.

"Effective, Affordable, Reusable Speech-to-Text (EARS)." *Effective, Affordable, Reusable Speech-to-Text (EARS)*. Electronic Frontier Foundation, n.d. Web. 05 Nov. 2015. <https://w2.eff.org/Privacy/TIA/ears.php>.

EpiCollect. Computer software. *EpiCollect.net*. Wellcome Trust, n.d. Web. 30 June 2015. <http://www.epicollect.net/>.

Flannigan, James L. "Techniques for Speech Analysis." Speech Analysis Synthesis and Perception. N.p.: Springer-Verlag Berlin Heidelberg, 1972. 166-67. Print.

"Global Autonomous Language Exploitation." *Global Autonomous Language Exploitation*. The Idiap Research Institute, n.d. Web. 05 Nov. 2015. <http://www.speech.sri.com/projects/GALE/>.

GOOG-411 Team. "Goodbye to an Old Friend: 1-800-GOOG-411." Web log post. *Official Google Blog*. Google, 8 Oct. 2010. Web. 5 Nov. 2015. <https://googleblog.blogspot.com/2010/10/goodbye-to-old-friend-1-800-goog-411.html>.

Hartzler, Bruce. *IDig*. Computer software. *Apple App Store*. Vers. 5.0.2. N.p., 12 Feb. 2015. Web. 30 June 2015. <https://itunes.apple.com/us/app/idig-recording-archaeology/id953353960?mt=8>.

Hartzler, Bruce. IPad Screenshot of iDig App. Digital image. *IDig - Recording Archaeology*. ITunes App Store, n.d. Web. 30 June 2015. <https://itunes.apple.com/us/app/idig-recording-archaeology/id953353960?mt=8>.

Hey You, Pikachu! Redmond, WA: Nintendo, 1998. Computer software.

IBird Pro Guide to Birds. Computer software. *Apple App Store*. Vers. 7.22. Mitch Waite Group, 15 Mar. 2015. Web. 30 June 2015. <https://itunes.apple.com/us/app/ibird-pro-guide-to-birds/id308018823?mt=8>.

"IBM Shoebox." *IBM Archives*. N.p., n.d. Web. 30 June 2015. <https://www-03.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html>.

Kempelen, Wolfgang Von, Heinrich Füger, and J. G. Mansfeld. *Wolfgangs Von Kempelen K.k. Wirklichen Hofraths Mechanismus Der Menschlichen Sprache: Nebst Der Beschreibung Seiner Sprechenden Maschine*. Wien: J.V. Degen, 1791. Print.

Klatt, Dennis. "The Klattalk Text-to-speech Conversion System." *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82*. Vol. 7. 1589-592. *IEEE Xplore*. IEEE. Web. 29 Sept. 2015. <http://dx.doi.org/10.1109/ICASSP.1982.1171431>.

Krishnaraj, Arun, Joseph K. T. Lee, Sandra A. Laws, and T. Jay Crawford. "Voice Recognition Software: Effect on Radiology Report Turnaround Time at an Academic Medical Center." *American Journal of Roentgenology* 195.1 (2010): 194-97. *American Journal of Roentgenology*. Web. 25 Nov. 2015. <http://www.ncbi.nlm.nih.gov/pubmed/20566816>.

LeafSnap. Computer software. *LeafSnap*. Vers. 1.07. Columbia University, University of Maryland, and Smithsonian Institution, 5 June 2015. Web. 30 June 2015. <http://leafsnap.com/>.

Lowerre, Bruce, and Raj Reddy. "The Harpy Speech Understanding System." *Readings in Speech Recognition* (1990): 576-86. Web. <http://dl.acm.org/citation.cfm?id=108277>.

Magpi. Computer software. *Apple App Store*. Vers. 3.0.3. DataDyne Group LLC, 22 June 2015. Web. 30 June 2015. <https://itunes.apple.com/us/app/magpi/id956798146?mt=8>.

Novet, Jordan. "Google Says Its Speech Recognition Technology Now Has Only an 8% Word Error rate." *VentureBeat*. N.p., 28 May 2015. Web. 30 June 2015. <http://venturebeat.com/2015/05/28/google-says-its-speech-recognition-technology-now-has-only-an-8-word-error-rate/>.

"Nuance Dragon Naturally Speaking, Medical Transcription, Voice Recognition Software." *Nuance Dragon Naturally Speaking, Medical Transcription, Voice Recognition Software*. N.p., n.d. Web. 30 June 2015. <http://www.nuance.com/dragon/index.htm>.

"One Handheld Per Breeder." *One Handheld Per Breeder.* McKnight Foundation, n.d. Web. 04 Nov. 2015. <http://www.wheatgenetics.org/research/one-handheld-per-breeder>.

Plant-O-Matic. Computer software. *Apple App Store*. Vers. 1.2. Ocotea Technologies, LLC, 9 Apr. 2015. Web. 30 June 2015. <https://itunes.apple.com/us/app/plant-o-matic/id906932765?mt=8>.

Project Noah. Computer software. *Apple App Store*. Vers. 2.6.1. Networked Organisms, 24 Feb. 2012. Web. 30 June 2015. <https://itunes.apple.com/us/app/project-noah/id417339475?mt=8>.

Rosen, George. Dynamic Analog Speech Synthesizer. George Rosen, assignee. Patent 3042748. 3 July 1962. Print.

Qayyum, Ali. Coffely interface. Digital image. *40 Beautiful Flat UI Design Inspiration*. Smashing Hub, n.d. Web. 23 June 2015. <http://smashinghub.com/40-beautiful-flat-ui-design-inspiration.htm>.

Shewry, Peter R. "Wheat." *Journal of Experimental Botany* 60.6 (2009): 1537-553. *Journal of Experimental Botany*. Oxford Journals. Web. 29 Sept. 2015. <http://dx.doi.org/10.1093/jxb/erp058>.

"Speech Synthesis—Invented by Christian Gottlieb Kratzenstein." *Edubilla.com*. EduBilla, n.d. Web. 30 June 2015. <http://www.edubilla.com/invention/speech-synthesis/>.

"Staple Foods: What Do People Eat?" FAO Corporate Document Repository. Food and Agriculture Organization of the United Nations, n.d. Web. 29 Sept. 2015. <http://www.fao.org/docrep/u8480e/u8480e07.htm>.

Stevens Creek. TripLog interface. Digital image. Learning from "bad" UI. Signal v. Noise, 9 July 2008. Web. 23 June 2015. <https://signalvnoise.com/posts/1128-learning-from-bad-ui>.

"Stewart's Electrical Analog." *Stewart's Electrical Analog*. Haskins Laboratories, n.d. Web. 30 June 2015. <http://www.haskins.yale.edu/featured/heads/SIMULACRA/stewart.html>.

Stratonovich, R. L. "Conditional Markov Processes."Theory of Probability & Its Applications 5.2 (1959): 156-78. Society for Industrial and Applied Mathematics. Web. 22 Dec. 2015. <http://epubs.siam.org/doi/abs/10.1137/1105015>.

"The Pattern Playback." *Haskins Laboratories*. N.p., n.d. Web. 30 June 2015. <http://www.haskins.yale.edu/featured/patplay.html>

There Came an Echo. *Steam App Store*. Vers. 1.0.5. Iridium Studios, 24 Feb. 2015. Web. 30 June 2015. <http://store.steampowered.com/app/319740/>.

Umeda, N., and R. Teranishi. "The Parsing Program for Automatic Text-to-speech Synthesis Developed at the Electrotechnical Laboratory in 1968." *IEEE Transactions on Acoustics, Speech, and Signal Processing IEEE Trans. Acoust., Speech, Signal Process.* 23.2 (1975): 183-88. *IEEE Xplore Digital Library*. Web. 29 Sept. 2015. <http://dx.doi.org/10.1109/TASSP.1975.1162663>.

Zickuhr, Kathryn, and Lee Rainie. "E-Reading Rises as Device Ownership Jumps." *Pew Research Center Internet Science Tech RSS*. Pew Research Center, 16 Jan. 2014. Web. 23 June 2015. <http://www.pewinternet.org/2014/01/16/e-reading-rises-as-device-ownership-jumps/>.