VOL. 13, NO. 7, APRIL 2018 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

DEVELOPMENT OF A SPEECH CONTROLLED ANDROID DRAWING APPLICATION FOR HANDICAPPED PEOPLE

Lukas Klinghammer[1], Muhammad Akbar Ibnu Farhan[2] and Wansu Lim[3]
[1]Department of Software Engineering, University Heilbronn, Germany
[2]Department of Electrical Engineering, Telkom University, Bandung, Indonesia
[3]School of Electronic Engineering, Kumoh National Institute of Technology, South Korea
E-Mail: wansu.lim@kumoh.ac.kr

ABSTRACT

A big margin of the handicapped people still do not have the same possibilities to express themselves and further advance their personality as the majority of the population. The control and use of smartphones proves to be a big challenge for them. Applications controlled by speech serve as a substitution to the finger controlled based applications that dominate the market. Through the use of such applications the handicapped have the chance to experience the life of the unburdened. Based on those implications the development of an application to support the artistic needs of the handicapped was imminent. The developed application achieves the support by being fully controllable by speech. The development proved to be a possibility but the limiting factors of the mobile phones and the pocketsphinx library hindered any groundbreaking results. Nonetheless the application allowed for a base research on the still small field of accessibility applications.

Keywords: speech recognition, android, mobile, smartphone, drawing, pocketsphinx, handicapped, application.

1. INTRODUCTION

The everyday life of people with disabilities is a struggle they must cope with. They are not able to use smartphones in a way others can. This proves to be a limiting factor in the possible activities they can engage in. To make life easier there are a lot of ways to support them with the use of modern technology, such as the use of voice commands in their homes. The support for people with special needs has grown in the last years through the development of new hardware and software [1].

People who cannot use their hands or arms have a hard time controlling smartphones and mobile devices in general. The voice is a natural tool most of the people can use to communicate with other people but also with electronic devices. To cater the need to artistically express themselves speech recognition controlled drawing applications can provide the foundation for the further.

The research on speech recognition on Android devices has come a long way. The biggest challenges stem from the influence of noise disruption and the shortage of processing and battery power [2, 3]. Additionally, the microphones of most mobile devices are not on the professional level that you would need to record speech in a sufficient quality [4]. Despite these factors the development of user created content stepped continually forward. Even though the mobile devices are not able to reach the level of professional speech recordings the quality is adequate for small applications like the control of household devices [5].The possibilities to use speech recognition as a tool to control Android applications is the focus of this research. Pocketsphinx [6] provides an easy to use library which enhances the potential of an application [7] to help the people with special needs.

In this paper, the use of an android speech recognition application to help handicapped people draw simple pictures is being looked at. By speaking the keyword “draw” users can activate the speech recognition and give commands like up, down, left etc. to draw a line on the screen. Through this method, the disabled can experience a new way of expressing themselves that was not possible before without speech recognition. Furthermore, this paper is going to discuss the possible expansion on the topic and how it can be used in more advanced scenarios.

2. METHODOLOGY

A. Android

Android is a software environment for mobile devices developed by Google Inc. [8]. Android is based on java and thus is suitable for object oriented programming. The applications are separated into different activities which can be split into fragments to display smaller parts of the activity. The ability to use bound services is the reason Android is the chosen operating system. These services are bound to an activity and can be used to broadcast information to the further. In our case the speech recognition runs in the background and we use the broadcast to send the recognized text to our drawing activity. Another advantage of Android is the wide supply of open - source and freely distributed software. One of those libraries is the speech recognition API CMU Sphinx developed by Carnegie Mellon University.

B. CMU Sphinx

The CMU Sphinx API allows for an easy use of a free speech recognition API [9]. The pocketsphinx package is developed for the use on mobile devices of all sort and focuses on low source platforms. Because of those reasons, it’s the perfect tool for our application. The speech recognition is based on similar sounds of the spoken language. These sounds are called phones and they are being matched to an existing language model, e.g. the English language. The speech recognizer uses different

grammar models to compare them to the complete library. We developed a grammar that is useful for the drawing application mainly consisting of commands like up, down, left, right etc.

3. APP LAYOUT

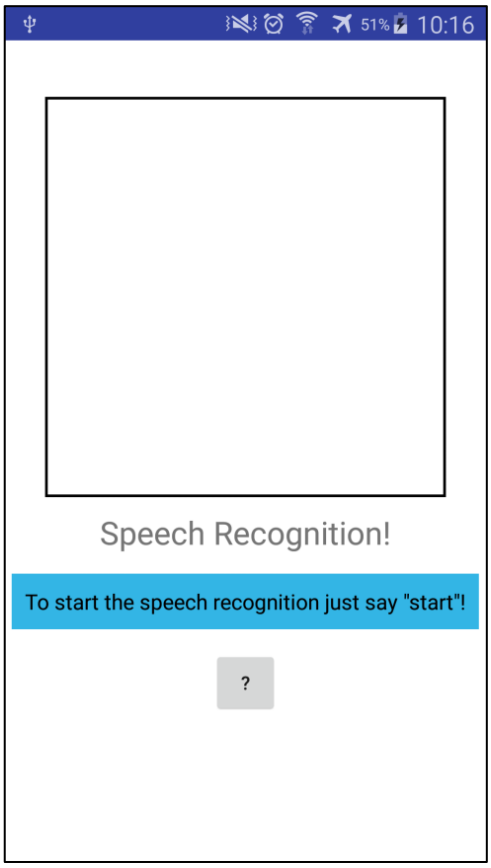

Figure-1. Layout of the Main Activity.

In the upper half of our application, as shown in Figure-1, the canvas is located. On this canvas, all the drawing done by the user is shown. When a new command is being recognized the canvas updates accordingly and displays a line. Directly below the canvas the actual command is shown to the user so that he can check if the service recognized the right command. When the app is first started the screen welcomes the user with a small text saying "Speech Recognition!" In the blue box on the bottom half a small manual is presented to the user. The text encourages the operator of the application to start the speech recognition by saying the keyword "start". This keyword is used every time the user wants to give a new command. Just below the manual a small button with a question mark on top is displayed. Whenever the user presses the button a help dialog is shown where the user can see all the available commands.

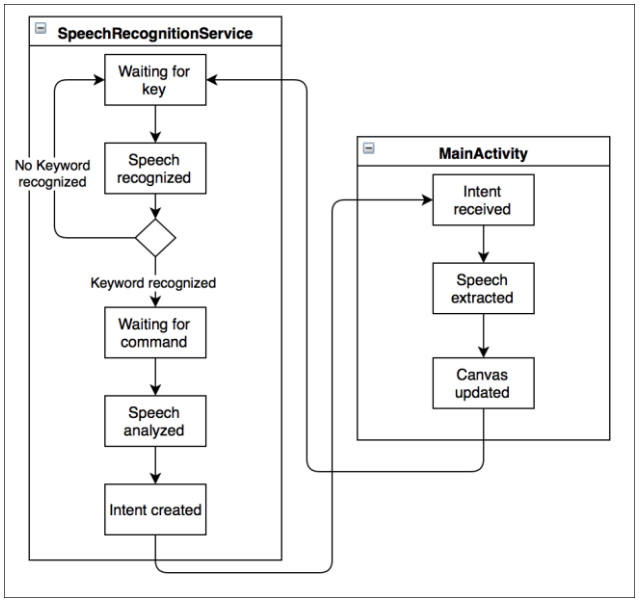

Figure-2. Workflow of the application.

4. DEVELOPMENT PROCESS

A. Service

The MainActivity (MA) class hold the visible part of our application. It shows the drawing and acts as a broadcast receiver. When the MA is created, it is going to bind a Speech Recognition Service (SRS) to itself by creating a new intent. The intent is created in the on Create() method of the MA. With this intent, it is going to establish a connection between itself and the service. When the on Bind() method of the service is called the speech recognition is going to be initialized by calling the run Recognizer Setup() method. The setup uses an asynchronous task to not block the other thread of the MA. The setup can be a time-consuming process that involves a lot of input and output handled by the pocketsphinx library. In the background the task will call the setup Recognizer() method to set the options for the recognizer. In this step the SRS loads the acoustic model, the dictionary, and the directory with the assets into the recognizer. In this case, we use the English language as the basis of our speech recognition. After the initial setup, the key phrase "start" is added so that the service can react to the input of the user. In the add Grammar Search() method the grammar created especially for the drawing application containing all the possible commands is provided to the recognizer. After the task is executed the service is going to switch into the search state to wait for the key phrase to be spoken. The service is constantly running in the background and there for provides the basis functionality of the application. Because of this the user can draw on the canvas without the press of a button.

```
methodonResult(hypothesis) {

if (hypothesis is notnull) {
  get text from hypothesis;

if (text is not keyphrase) {
    create new Intent;
    add text as extra to intent;
    send Intent to broadcast manager;
    }
}
```

Figure-3. Pseudocode of the on Result () method.

B. Broadcast

The communication between the MA and the SRS is established by a broadcast system. In the on Create() method of the MA a local broadcast manager is established that listens to intents with the filter "speech_recognized". Whenever the SRS calls the onResult() method, after recognizing a spoken word of the defined grammar, it is creating a new intent with the "speech_recognized" subject. As an extra it is going to add the actual word that was recognized. The broadcast manager of the MA receives this intent. By calling the get String Extra() method it extracts the command out of the intent and calls the drawLine() method to display the line on the canvas.

C. Pocket sphinx

The runRecognizerSetup() method of the SRS sets the basic options of the pocketsphinx speech recognition. While setting up the service adds the English dictionary and the English acoustic model to the recognizer so that it can understand the English language. In the next step the key phrase "start" is added to provide the user with the ability to start the drawing at his will. After adding the key phrase the grammar for the drawing application is made available to the recognizer. The function of the grammar will be further explained in the *E. Grammar* section. After the setup, the SRS is swapping into the keyword search mode. In this mode, the service is waiting for the user to speak the keyword "start". When the key phrase is recognized the switch into the drawing search is performed. The first recognized word is compared to the grammar. The grammar consists of all the words our application actively uses. Such a grammar is developed to narrow down the complete English language to a few distinct keywords. In the definition of the grammar. If it is an element of the grammar the speech recognition is going to stop and onResult() is called. After the intent is sent it will go back into the keyword search and wait for the user to draw the next line.

D. Canvas

For the application to be able to display the user commands it must be able to draw on the screen. The canvas fulfills this role and is being setup up when the onCreate() function of the MA is executed. In setupCanvas() the background bitmap is initialized and added onto the canvas. In this process the start coordinates are set as well to the middle of the drawing board. Whenever the broadcast manager calls the draw() method the canvas updates itself to show the new content. The canvas allows for different commands which can be executed to create lines or shapes on the bitmap. This functions include drawLine() to draw simple lines from startX and startY to endX and endY, drawRect() to create a rectangle from left to top to right to bottom, drawCircle() to draw a circle around the center with a certain radius, and clear() to remove any drawings from the bitmap. To display different line strengths and colors the canvas uses a Paint object to set the necessary information. The setColor() method sets to color to a desired color, set Stroke Width() uses a input pixel size to determine the width of the lines, and set Style() defines if the shapes will be filled or just stroked.

```
methodonReceive(intent) {

get text from intent;
set text of the textview;
call draw(text) to draw the command;
```

Figure-4. Pseudocode of the onReceive() method.

E. Grammar

Because of the huge amount of available words in the English language a grammar designed for the individual usage must be created. The grammar describes all the possible words the speech recognition can decipher out of the recognized phones. The java speech grammar format is used which allows for complex files. The grammar is separated into different packages and name spaces. After the name is declared as many rules as the user needs can be added by using the defined standards in the JSpeech rules. A new one can be opened by using for example public <direction> or public <shape>. Such a grammar is essential for the speech recognition to perform in an efficient and performant way.

5. ALGORITHM

The Algorithm works as explained in the following chapter and is shown in Figure-2. When the application is started, the SRS is going to wait for the keyword "start" (Waiting for key). As soon as the SRS recognizes a word it will compare the recognized speech to the underlying keyword (Speech recognized).

a) Waiting for key and speech recognized

The recognizer calls onPartialResult() every time he recognizes a new word. This word is compared to the key phrase "start". If the words do not match the recognizer just continues to listen to whatever is incoming and comparing it to the key phrase. If the speech is indeed the keyword "start" the service switches into the drawing search and uses the defined grammar to wait for a feasible command to execute.

After the keyword was accepted the speech recognition awaits the next input by the user (Waiting for command). The next word is compared to the drawing grammar (Speech analyzed). The service creates an intent and adds the word as an extra to transfer it to the MA (Intent created).

b) Speech analyzed and Intent created

When the SRS receives a result it is going to call the onResult() method as shown in Figure-3. The function is checking the incoming result is not null to make sure that there is a result to be sent to the MA. Afterwards it extracts the actual speech out of the Hypothesis object so it can add it to the intent. The now extracted speech is compared to the keyword to prevent the keyword from being sent to the broadcast. After all the comparisons have been successful the intent is created and the recognized text is added as an extra. Now the intent is sent to the broadcast manager.

The broadcast manager of the MA catches the intent and processes the information held by it (Intent received). Following in the next step the recognized word is extracted from the intent (Speech extracted).

c) Speech extracted

When the broadcast manager receives the intent, as shown in Figure-4., it will first extract the text out of the extra of the intent. Now the function calls the set Text() method of the text View to display the recognized speech to the user. The extracted text is used as a parameter to call the draw() method of the MA to show the new drawing on the canvas.

The extracted speech is sent to the draw method. Depending on the command the canvas of the MA is going to be updated (Canvas updated).

d) Canvas updated

The draw() method of the MA, as shown in Figure-5, receives the recognized text to be converted to a drawing. A new Paint object is created to display the lines with a thickness of 2 pixels and a black color. All the shapes will be filled with the paint. Depending on the incoming text the switch cases uses the appropriate case to show the change. The switch case is either drawing lines, a rectangle, a square, a circle, clear the canvas from all drawings, set the color or show the help dialog.

```
methoddraw(text) {
switch (text) {
case“up”: draw line upwards;
case “down”: draw line downwards;
case “left”: draw line to the left;
case “right”: draw line to the right;
case “rectangle”: draw rectangle;
case “square”: draw square;
case “circle”: draw circle;
case “red”: set color to red;
case “blue”: set color to blue;
case “green”: set color to green;
case “black”: set color to black;
case “clear”: clear canvas, reset color;
```

Figure-5. Pseudocode of the draw() method.

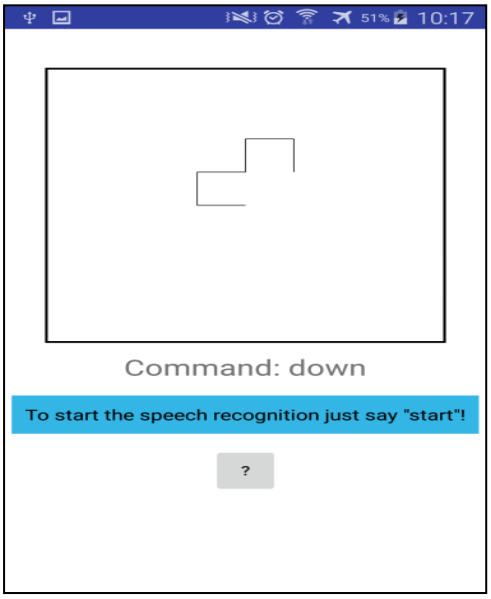

Figure-6. Six lines drawn on the canvas.

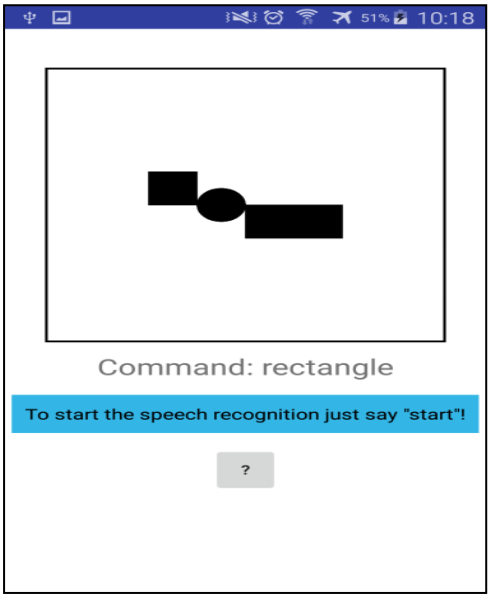

Figure-7. Different shapes displayed on the canvas.

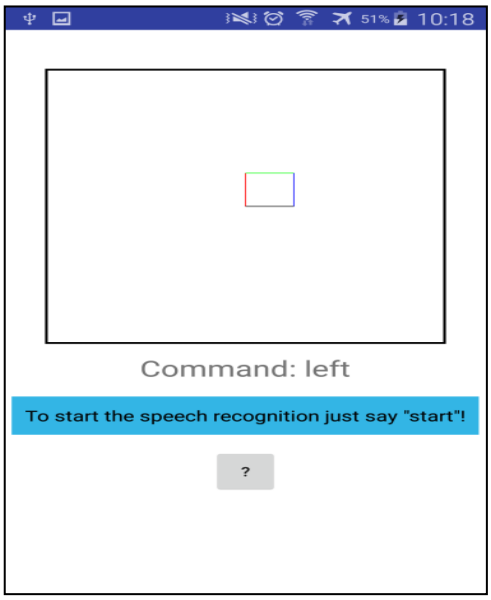

Figure-8. The four different colors.

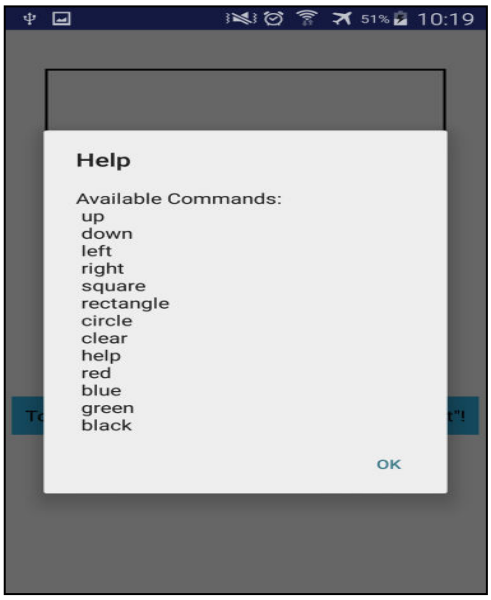

Figure-9. The help dialog.

After the drawing function is completed the SRS service waits for the keyword to begin the process again (Waiting for key). This whole workflow can be repeated indefinitely.

6. RESULTS

In Figure-6 the execution of a command chain to draw on the canvas is shown. The lines are all connected and thus allow the user to create any image. The recognized commands where left, up, right, up, right, and down. In the next step the clear order was spoken and the application erased all the drawn lines from the canvasso that the user can start to draw a new picture. In Figure-7 the different shapes are displayed. Firstly, three lines were drawn to get to a position that was on the left side of the canvas. After that the shapes were drawn in the following order: square, circle, and rectangle. The user also has the ability to choose different colors as shown in Figure-8. A square consisting of four lines with the colors red, green, blue and black were drawn consecutively.

By either pushing the question mark button or saying help the user can open a small help window, as shown in Figure-9, to guide him through the application. In this window, all the available commands are shown. Either after five seconds or by a push on the "OK" button the popup closes itself.

When it comes to performance the application is a bit lacking. The limiting factors of the small CPU power and the tuned down range of functions of the pocketsphinx result in a slow work process. The time it takes for the speech to be recognized, analyzed and shown on the canvas proves to be too long for a professional use. Nevertheless, the basic functionality works fluently and showed no imminent bugs.

CONCLUSIONS

As shown in the paper the possibility for the use of speech recognition to help handicapped people artistically express themselves is given. Such an app caters the self - actualization needs of the Maslow's Hierarchy of Needs [10] and thus brings joy and motivation in the life of the user. The restrictive hard- and software prevented the development of an application on a bigger scale. Such could be done on a bigger and more powerful system like a tablet or a notebook. On one of those systems the application could be further expanded to even help handicapped people effectively do jobs they normally could not, like product design or architecture.

ACKNOWLEDGEMENTS

This work was supported by the Global Excellent Technology Innovation Program (10063078, Development of Textile based Wearable Input Auxiliary Device and UI system for the Disabled) funded by the Ministry of Trade, Industry and Energy (MOTIE) of Korea.

REFERENCES

[1] S. A. F. Manssor, A. A. Osman and S. D. Awadalkareem. 2015. Controlling Home Devices for Handicapped People via Voice Command Techniques. in International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering, Khartoum, Sudan.

[2] K. Garg and G. Jain. 2016. A Comparative Study of Noise Reduction Techniques for Automatic Speech Recognition Systems. in Intl. Conference on Advances in Computing, Communications and Informatics, Jaipur.

[3] A. Jayakumar, M. Raghunath, S. M.S, A. S, A. Sadanandan and P. Nedungadi. 2016. Enhancing Speech Recognition in Developing Language

Learning Systems for Low Cost Androids. in International Conference on Computational Techniques in Information and Communication Technologies, New Delhi, India.

[4] P. Mittal and N. Singh. 2016. Speech based command and control system for mobile phones: Issues and Challenges. in Second International Conference on Computational Intelligence & Communication Technology, Ghaziabad.

[5] N. Mulhern, N. McCaffrey, N. Beretta, E. Chabot and Y. Sun. 2013. Designing Android Applications using Voice Controlled Commands for Hands free interaction with Common Household Devices. In: 39th Annual Northeast Bioengineering Conference, Syracuse.

[6] D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar and A. I. Rudnicky. 2008. POCKETSPHINX: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices. IEEE, Pittsburgh.

[7] D. S. Ganesh and D. P. K. Sahu. 2015. A Study on Automatic Speech Recognition Toolkits. in International Conference on Microwave, Optical and Communcation Engineering, Bhubaneswar.

[8] Android Developer. 2016."Google, [Online]. Available: https://developer.android.com/index.html. [Accessed 18 11 2016].

[9] CMU Sphinx. 2016. [Online]. Available: http://cmusphinx.sourceforge.net/. [Accessed 17 11 2016].

[10] J. E. Gawel. 1997. Herzberg's Theory of Motivation and Maslow's Hierarchy of Needs. ERIC Publications, Washington, DC., USA.