Human Whistle Interface

Patricio Córdova
University of Toronto
patricio@cs.toronto.edu

***Abstract*—Speech is an interface that can boost the ease of use of smartphones and that is beginning to be widely used around the world. Applications like Siri, Google Now, or Cortana are some examples [1]. However, these are tools that haven't been perfected yet [2]. Looking for an alternative method to use sounds produced by humans to communicate with a smartphone, the idea of using a sound that is more universal, easy to produce and easy to detect arose. This paper explores the use of the whistle as an interface to trigger actions in a smartphone. We describe the methodology used to detect a whistle, how this methodology was implemented in an Android application, how all the components of the application were developed, the application resulting from this work, and some experimental results.**

***Index Terms*—application software, audio user interfaces, automatic speech recognition, human computer interaction, smart phones application, smart phones, sound recognition, speech analysis, speech recognition, user interfaces.**

I. Introduction

Speech recognition is a computer science area that treats the problem of the translation of spoken words into text.

Speech recognition applications include voice user interfaces such as voice dialing (e.g. "Call home"), call routing (e.g. "I would like to make a collect call"), domestic appliance control, search (e.g. find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), preparation of structured documents (e.g. a radiology report), speech-to-text processing (e.g., word processors or emails), aircraft (usually termed Direct Voice Input), among others.

If we consider the case of smartphones in particular, voice user interfaces would considerably speed up the way we interact with our phones. Given the screen size of today's smartphones (~4.2 – 5.5 inches) and all the possible actions that users trigger on them every day (i.e. watching videos, chatting, answering emails, checking social networks, etc.); touch-based interfaces are not the most optimal: we are limited to use a screen of small size and we are forced to use two of our senses (sight and touch) to use the device [3].

If voice user interfaces were completely accurate, meaning this that the interface is able to recognize every word that we pronounce, we could trigger almost any of the actions that we use in our smartphones every day, like browsing the internet, checking social networks, playing games, listening to music, making calls, checking/writing emails, sending text messages, watching clips, taking photographs, disabling our alarm clock, and others. If speech recognition were accurate enough, we could order our phone to do whatever we want. However, it is a tool that hasn't been perfected yet.

The best commercial and general public available efforts to create systems that can efficiently recognize human words are Siri and Google Now. According to Piper Jaffray's research in a study [4] made with 800 queries to Siri and Google Now, Siri's ability to translate speech in a noisy environment has an accuracy of 91% in iOS7, whereas Google Now revealed an accuracy of 88%.

Nevertheless, if we consider that 10% error and other factors like that the fact that the study made by Piper Jaffray's research doesn't reveal the accuracy for foreign people or people with speech disorders [5] speaking directly at the phone, it is still a tool that cannot reach the efficiency of haptic interfaces [6] for instance.

This paper talks about the use of a more generic human sound as an attempt to use it to trigger simple actions in a phone, sound that can overcome some of the barriers of speech recognition like accuracy, different accents, and speech problems. This sound is more uniform, can be louder than speech, and can be more easily understood by a machine. It is the whistle.

In fact, the whistle can serve as a language itself. The Silbo Gomero (Gomeran whistle), which is a whistled language spoken by inhabitants of La Gomera in the Canary Islands, is used to communicate across the deep ravines and narrow valleys (gullies) that radiate through the island. Due to the mountains, communication was very difficult among shepherds, and that is the reason why a new way of communication was born.

Although, this is not the case and we are not going to use a whistle as extensively (because it is obvious that a whistle cannot replace speech or even be compared to it), the proposal of this document consists only in using this sound to trigger very simple actions in determinate contexts, like when it is more useful to use one single voice command and it is not possible to use speech.

In the next section we will discuss more in depth the way that this interface designed and implemented.

II. Objectives

The goal of this work was to design an application that can be operated by whistles. After some deliberation, the options were: a media player that can be controlled by whistles when people are driving or running, an

application that can increase/decrease the brightness or volume of the phone depending on the tone of the whistle, a camera for which the trigger is a whistle, and a phone locator that can help you locate your phone by whistling at it.

The option chosen was the last one. First, we are going to assume that the phone will be capturing and analyzing all the sound that comes to the microphone so that when the user whistle to the phone, the phone understands the whistle and respond at it in a specific way.

III. Whistle Detection

The technique used for whistle detection is based on the following steps:

1. The sound wave is captured in small batches over time.
2. A window function is applied to each wave in order to reduce the spectral leakage. The reason will be explained further.
3. The Fast Fourier Transform (FFT) is applied to the sound wave in order to convert the signal from the time domain to the frequency domain.
4. Statistical calculations are performed over the resultant signal to determine if it corresponds to a whistle. These calculations are performed over the following minimum/maximum parameters: frequency, intensity, standard deviation and zero crossing rate.

As can be seen, the fourth step for the whistle recognition relies on statistical methods. This means that only sounds that have special characteristics in frequency or intensity (like whistles, claps, fingers clashes, etc.) can be detected using this methodology.

Fig. 1. Initial design of the frontend.

IV. Application Design and Implementation

Once the previous parameters were decided, the next step was to design the application itself. The design of the graphical user interface of the application, that will launch the service which will listen to the whistles, consisted in creating a process relatively simple: one button to make the app listen and another button to close the app. The first mockup can be seen in Figure 1. In the right side of Figure 1 there is a menu that was included in the first design, however, it was not implemented because of limitations of the device and also because this was a feature that was considered not useful.

A. Whistle Detection Implementation

The whistle detection module is the heaviest component of the application. It was implemented using a sound recognition package [7] contributed in its majority by Jacket Wong and Sun Microsystems (FFT).

Recalling the steps previously presented in Section III, the first step is to capture the sound waves that come to the phone's microphone. For this sake the Android class used is AudioRecord. This class gets the sound and encodes it using the following configuration parameters: sample rate, channel configuration, audio encoding, and size of the buffer (interval in which a batch of bytes are considered to be processed as block of sound). The values used for these parameters are the following:

- Sample rate: 44100.
- Channel configuration: MONO.
- Audio encoding: Pulse-code modulation of 16 bits.
- Buffer size: 16 bits.

These parameters comprise most basic sound encoding. This makes the sound analysis faster if we consider that all the methodology previously explained to detect the whistle has to be applied to every single buffer of 16 bits.

Next, a windowing function is applied to the signal prior applying the FFT. FFT computation assumes that a signal is periodic in each data block, that is, it repeats over and over again and it is identical every time. When the FFT of a non-periodic signal is computed then the resulting frequency spectrum suffers from leakage [8]. Leakage results in the signal energy smearing out over a wide frequency range in the FFT when it should be in a narrow frequency range. As the signal captured by the phone is not periodic in the periods when it is obtained, a window function must be applied in order to correct the leakage and leave the signal ready to apply the FFT. A window is shaped so that it is exactly zero at the beginning and end of the data block and has some special shape in between. This function is then multiplied with the time data block forcing the signal to be periodic. FFT windows reduce the effects of leakage but cannot eliminate leakage entirely. In effect, they only change the shape of the leakage. There are different types of window functions depending on the type of the signal received. In this case the windowing function chosen was "Hamming" because it is more appropriate for processing 8/16 bits audio signals.

After that, the next step consists in applying the FFT to the signal. The FFT is an algorithm to compute the discrete Fourier transform (DFT) and its inverse. Fourier analysis converts time (or space) to frequency and vice

versa [9]. Figure 2a represents a complex signal in the time domain. Figure 2b is the result of running the FFT over the first signal. The Fourier transform outputs a histogram of the intensity of each frequency registered. It is exactly what allow us to detect a whistle in the wave: if the FFT resultant shows that the range of frequencies corresponding to a whistle are present, then we have recognized a whistle.

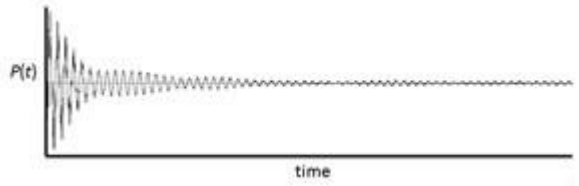

Fig. 2a. Signal in the time domain.

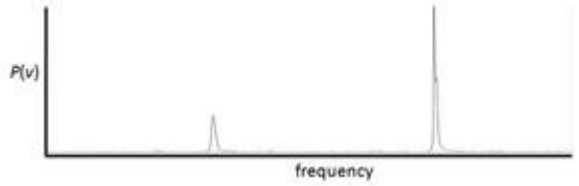

Fig. 2b. Signal in the frequency domain. Result of the FFT.

Finally, the last step is to determine if there is a whistle in the signal obtained from the FFT. A human whistle is typically located in the range of 500-5000Hz [10] [11]. However, some people can exceed this limits, so the upper boundary considered for the application was the highest number allowed by the programming language. These are the most important ranges for finding the whistle. The last parameters to be considered are the min/max values for the statistical tests. The values considered were:

- Intensity: 100Hz – 10'000Hz
- Standard deviation: 0.1 – 1
- Zero crossing: 50 – 200

These were the values that provided most accurate whistle recognition. However, as you will see in the results section only 80% of the whistles were recognized (having the phone at one arm of distance). We expect to perform more proof-error experimentation in the future to improve this performance.

B. Application Implementation

To create the actual application, there were some elements that had to be put together:

1. Application frontend: The frontend consists in an Android Activity (screen) that contains the button to launch and stop a service that will be in charge of detecting the whistle.

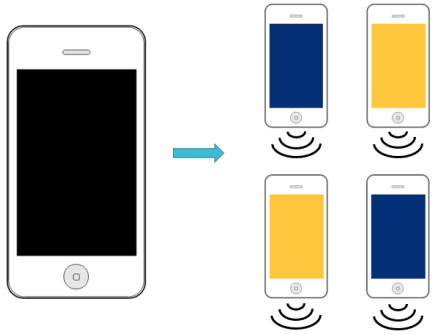

Fig. 3. Application's reaction to whistle.

2. Listener service: Once the "listen" button is pressed by the user, there is an Android "Sticky" service that is launched and that will be in charge of the following actions: record the audio that comes from the microphone, provide this audio to the whistle detection interface (configured to detect the whistle according to the methodology presented in the previous section), and finally react to the whistle's detection. A sketch of the concept can be seen in Figure 3. The reaction triggered will consist of 3 steps:
   2.1. The phone will play the chirp of a cardinal bird. This sound will be played after the whistle is recognized and will shut down after its termination (so that the sound doesn't keep playing indefinitely annoying the user). The user will have to whistle again in order to repeat the chirp.
   2.2. The phone will launch a screen with mutating and vibrating colors with the text: "Here I am!", and the phone will raise the brightness of the screen to the top value. This screen will keep being shown until the user gets the phone and deactivates the app.
   2.3. The flash light of the phone will be turned on while the sound is playing.

   The idea behind this series of actions is to help the user find the phone easily after the whistle. Another consideration is that in the first versions of the application the detection service ran in the background indefinitely having a relatively medium/high impact in the battery life (around 30% consumption). In the final version, the phone was set to listen only in when the phone is locked, reducing the battery impact (reduction to 22-24% consumption).

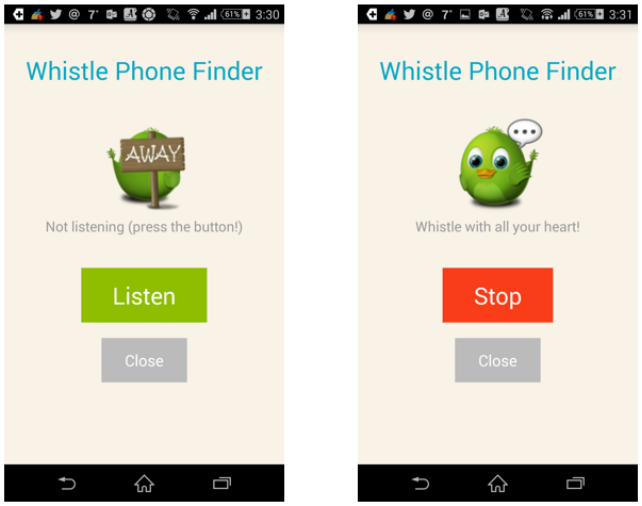

Fig. 4. Application's final design.

V. Results

The final version of the application frontend can be seen in Figure 4. The logo was chosen to resemble the chirp sound used as if there were a bird replying to the whistle.

After the app was developed and tested, there was a small survey done with 24 people to try the effectiveness of the whistle interface and the whole idea in general. The survey consisted in 6 questions and one try whistling at the app. The subjects were 24 graduate students of University of Toronto from the Faculties of Computer Science and Electrical Computer Engineering. The questions were:

1. Do you know how to whistle? Just after this, people had to whistle to the phone. Regularly most of them whistled at one arm of distance in a medium noisy environment (offices with people talking around and going in and out). It would have been more useful to try the app in a more real environment and from with a longest distance but the time of the students to complete the survey was very short.
2. Do you usually lose your phone in a closed place? A closed place means a room, or an office, or inside a bag.
3. Rating from 1 to 5, where 1 means completely useless and 5 means extremely useful, do you think that the app is useful?
4. Rating from 1 to 5, where 1 means completely useless and 5 means extremely useful, do you think that another similar interface, like a clap, would be more useful for the same purpose?
5. Do you like the app? This question is interesting too because we will see in the results that completely apart of the usefulness of the app, most of the people really liked the interface and the idea itself.

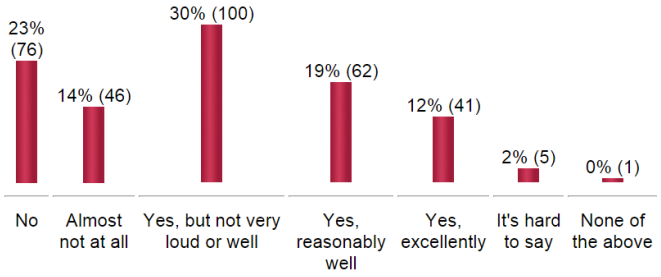

Fig. 5. People around the world that can whistle.

The results for the test and their interpretations go as follows:

1. From the 24 subjects studied, only 10 of them knew how to whistle. This represents the 40% of our participants. It is close to the results of a survey made to 331 people around the world by the site postyour.info [12]. The results of the postyour.info's survey can be seen in Figure 5. From the 10 subjects of our survey that know how to whistle, there was a count of how many times they had to whistle until the app recognizes their whistle. The results in Figure 6 show that 6 of them didn't have problems having their whistle recognized, 2 of them had some trouble (having to whistle more than 1 time), and 2 of them couldn't be recognized by the phone at all.

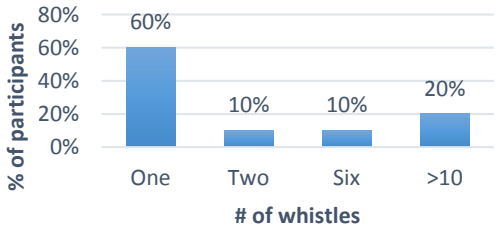

Fig. 6. Number of times the user had to whistle in order to have the application to recognize their whistles.

2. From the 24 subjects, 38% of them reported that they frequently lose their phones in closed spaces. This question shows a correlation with the next one.
3. In this question, the 33% of the users found extremely useful to have an app that can help them find their phones. If we compare this result to the previous one (where 38% of the users lose their phones), it can be seen that they are the group for which the app is very useful.

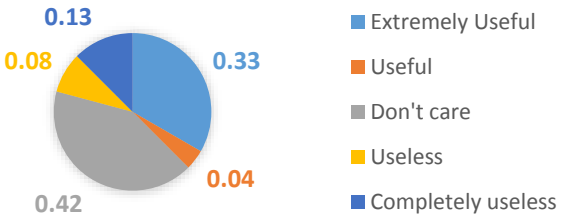

Fig. 7. Usefullness of the application.

4. This question shows if people prefer a whistle interface compared to another interface, in this case a similar one: the clap. The results are shown in Figure 8.

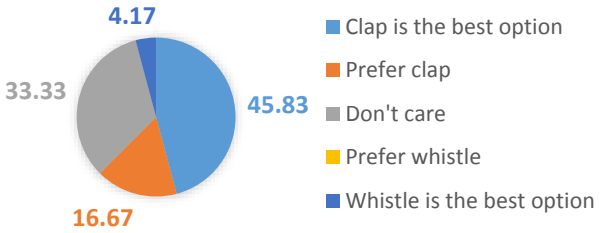

Fig. 8. Preference of clapping over whistling.

5. This question shows how “engaging” the users consider the application. It is interesting to see the results: while the app is useful for only around the 37% of the respondents, more than 80% of them consider it very engaging/fun. The results are presented in Figure 9

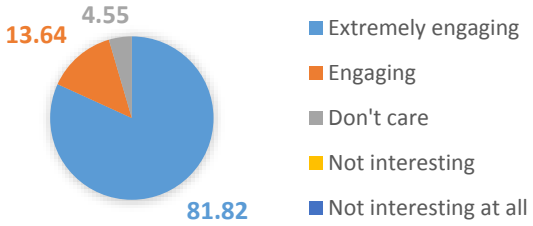

Fig. 9. How engaging the users consider the application.

VI. Conclusion

This paper discusses the implementation of a smartphone application that uses a whistle based interface. The whistle is a distinctive sound produced by humans that have its own frequency characteristics, and thus its recognition can be simpler than words or other sounds. In this document we cover the methodology used to detect the whistle, the design of an application that uses whistle recognition, and some experimental results.

The methodology can be resumed in the usage of the Fast Fourier Transform in order to convert the sound signal received by the phone from the time domain to the frequency domain and then apply some statistical methods to find the whistle.

The application developed consists in a “phone finder” system that can recognize whistles and that triggers certain actions in the smartphone after the recognition. It is basically used to locate a phone in a closed area.

The results of the experiment reveal that whistle recognition is viable as the application recognized around 80% of the subject’s whistles. However, as not everybody knows how to whistle, another similar interface (clap) was suggested by most of the respondents to be used instead/along with whistles. This will be considered for further work. Finally, although only 37% of the participants found the application useful, more than 80% of them consider the application entertaining. As a last step, there will be some tuning and testing of the application in order to upload it to the Android Play Store.

VII. References

[1] Pierre Lison and Raveesh Meena, "Spoken dialogue systems: the new frontier in human-computer interaction," *XRDS: Crossroads, The ACM Magazine for Students - Natural*, vol. 21, pp. 46-51, 2014.

[2] Markus Forsberg, "Why is Speech Recognition Difficult?," *Technical Report from Department of Computing Science Chalmers*, 2003.

[3] Dongsong Zhang, Hsien-Ming Chou, and Lina Zhou, "Hype or Ready for Prime Time?: Speech Recognition on Mobile Handheld Devices MASR," *International Journal of Handheld Computing Research*, vol. 3, pp. 40-55, October 2012.

[4] Chuck Jones. (2013, October) Forbes.com. [Online]. http://www.forbes.com/sites/chuckjones/2013/12/10/apples-siri-and-google-both-get-c-grades-from-gene-munster/

[5] Kinfe Tadesse Mengistu and Frank Rudzicz, "Comparing Humans and Automatic Speech Recognition Systems in Recognizing Dysarthric Speech," *Canadian AI'11 Proceedings of the 24th Canadian conference on Advances in artificial intelligence*, pp. 291-300, 2011.

[6] Niels Henze, Enrico Rukzio, and Susanne Boll, "100,000,000 Taps: Analysis and Improvement of Touch Performance in the Large," *MobileHCI '11 Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pp. 133-142, 2011.

[7] Jacquet Wong. (2012, July) musicg. [Online]. http://code.google.com/p/musicg/

[8] LDS Group, "Understanding FFT Windows," LDS Group, Manual 2003.

[9] Tony Dicola. (2014, July) Adafruit. [Online]. https://learn.adafruit.com/fft-fun-with-fourier-transforms/background

[10] Mikael Nilsson et. Al, "Human Whistle Detection and Frequency Estimation," *CISP '08 Proceedings of the 2008 Congress on Image and Signal Processing*, vol. 5, pp. 737-741, 2008.

[11] Gil Lopes, Antonio Fernando Ribeiro, and Paulo Carvalho, "Whistle sound recognition in a noisy environment," *Proceedings of Controlo'2010 - 9th Portuguese Conference on Automatic Control*, pp. 172–179, September 2010.

[12] (2014, November) postyour.info. [Online]. http://postyour.info/statistics/can-you-whistle.htm

[13] Ben Shneiderman, "The limits of speech recognition," *Communications of the ACM*, vol. 43, pp. 63-65, September 2000.

[14] Andrea Bonarini, Daniele Lavatelli, and Matteo Matteucci, "A Composite System for Real-Time," *RoboCup 2005*, pp. 130-141, 2006.