Advances on Breathing Based Text Input for Mobile Devices

Jackson Feijó Filho(✉), Wilson Prata, and Thiago Valle

Nokia Technology Institute, Av. Torquato Tapajós, 7200 – Col Terra Nova, Manaus, AM 69093-415, Brazil
+55 92 98134 0134
{jackson.feijo,wilson.prata,thiago.valle}@indt.org.br

Abstract. This paper highlights the progress of exploring a puffing activated keyboard for mobile phones. This approach aims to stand as an assistive technology for users with motor disabilities. From the implementation of prior versions we were able to identify recurring and persistent issues, such as ambient noise handling and keyboard layout. Some of these issues were detected during the experiments and some were reported by users. The advances achieved in this work are narrated from the outcomes of the implementation and experimentation of a mobile phone application that handles e.g. background noise by performing signal processing and a new keyboard layout.

1 Introduction

People with disabilities have restricted opportunities in a lot of areas, and the field of mobile technologies is no exception. A virtual environment offers the option to operate or to function in the real world, reducing physical boundaries. Technology users are likely to use their arms, hands, and fingers when using a mobile phone. Nonetheless, for people with motor disabilities, the need for movements that require dexterity can be a barrier for them to be able to interact with the phone [9].

Assistive technologies have been demanded, generally targeting users with severe motor disabilities such as motor neuron disease (MND), amyotrophic lateral sclerosis (ALS), spinal muscular atrophy, spinal cord injury, cerebral palsy, locked-in syndrome and Guillain-Barrè Syndrome (GBS) [8]. There are many types of physical disabilities that affect a pleasant human-computer interaction. As mentioned above, these diseases cause muscle deterioration and/or neurological disorders and consequently result in physical weakness, loss of muscle control, paralysis, and even amputation. In this context, the central issue for these users is the capacity to access computer controls (i.e. power/volume switch, menu items selection) and the aptitude to type on the regular keyboard or move pointing devices like a mouse cursor.

Assistive solutions have been investigated and established as an important field in human–computer interaction. Several forms of assistance have been produced for technology consumers with particular disabilities that lessen their capabilities to achieve inputting text on some devices. Several alternative interfaces for users with motor disabilities have been developed and reported. Frequently, these solutions include procedures that implement speech recognition methods, eye-trackers and

© Springer International Publishing Switzerland 2015
M. Antona and C. Stephanidis (Eds.): UAHCI 2015, Part I, LNCS 9175, pp. 279–287, 2015.
DOI: 10.1007/978-3-319-20678-3_27

sip-and-puff controllers. Speech recognition software is well accepted to be useful as textual input aid, while additional hardware are usually employed as pointing devices, allowing the control of e.g. the mouse pointer [2].

2 Related Work

Textual input and other interaction for people with motor disabilities and their phones have challenged the academy and industry to investigate alternative software and hardware solutions. These solutions will not depend on any manual interaction e.g. keystroking, screen-touching. Text input through other physiological signals [1, 4] is often considered. Speech recognition is also a solution used by the targeted audience.

2.1 Interaction Through Physiological Signals

Common physiological measures already used in previous Human Computer Interaction (HCI) and Accessibility works include: cardiovascular, electrodermal, muscular tension, ocular, skin temperature, brain activity and respiration measures [2]. These solutions typically require extra hardware, which makes these alternatives more expensive to final users. Not to mention they are ‘immobile’ computer oriented solutions (desktops, laptops, etc.) and do not cover mobile phones. Some works present sip-and-puff controlling that aims mobile devices, but requiring extra hardware.

2.2 Voice/Speech Recognition

Speech recognition is able to present itself as software based solution, which means no extra hardware is needed. But it affects the privacy of the user, as it implies the representation of whatever is being commanded through the form of speech to the periphery auditory. Consider the use case where one has to perform a phone call from within a bus. The user will obviously say out loud “call”. This may cause an unpleasant situation when the user is not willing to share e.g. the contact being called. Some hardware work has been developed to minimize peripheral representations of speech and improve privacy, as in [1] but again this requires not only extra and expensive hardware but the burden of wearing cochlear implants.

2.3 BLUI: Low-Cost Localized Blowable User Interfaces

BLUI [4] presents a form of hands-free interaction that can be implemented on laptop/desktop computing platforms. This approach supports blowing at a laptop or computer screen to directly control certain interactive applications. Localization estimates are produced in real-time to determine where on the screen the person is blowing. This work relies on a microphone, such as those embedded in a standard laptop or one placed near a computer monitor. Even though this approach is: very cost-effective, since no extra-hardware is required; silent and discrete, since no voice/speech is needed, it may not fit some mobile 158 computing platforms because: mobile phones need simpler computing processes due to limited processing power, if compared to laptops, desktops; several mobile phone interfaces are not based on (mouse) pointers.

2.4 PuffText

This is an earlier version of the present solution. It proposes the use of a low-cost software-based puff controlled hands-free spinning keyboard for mobile phones as an alternative interaction technology for people with motor disabilities. It attempts to explore the processing of the audio from the microphone in mobile phones to select characters from a spinning keyboard. A proof of concept of this work is demonstrated by the implementation and experimentation of a mobile application prototype that enables users to perform text input through "puffing" interaction.

2.5 Text Entry Using Single-Channel Analog Puff Input

The purpose of this prototype [11] is to demonstrate a use of puff input for text entry. The entry method is based on hierarchical scanning like selection of characters located in a static table organized according to the letter frequency. The cursor is moved in a way similar to operating a claw crane machine with two buttons. To move the cursor to the target position the user needs to produce two puffs, the first selects the column, and the second selects the row. With a brief training the method is capable of entry rate of 5 WPM.

This work has inspired the grid layout approach and the column-row, two step selection.

3 Advances on Breathing Based Text Input for Mobile Devices

From the several experiments and interviews performed during [10, 12], we were able to identify significant and recurrent issues concerning puff-based interfaces for mobile devices, some of which we aim to address in this work, as an evolution from [7].

3.1 Ambient Noise

In [12] the detection of a puff was being performed with a simple sound level peak, using the phone's microphone. Whenever the solution was tested on rooms with a lot of noise, a high number of false positives was detected. Even when the sound level threshold was adjusted, if e.g. someone spoke with a 68–76 dB level (normal voice speaking [16]) the software would trigger a puff.

To address the proper detection of the puffs, we used a Fast Fourier Transform (FFT) algorithm, as implemented in [3, 17]. This has also enabled a puff calibration feature, in order to adjust the puff detection to a more customized use.

3.2 Using FFT to Process Signal from the Microphone on Mobile Phones

In this investigation, we are targeting a Windows Phone 8.1 enabled device to implement the system and perform experiments. The code below shows part of the sound matching procedure:

```
void microphone_BufferReady(object sender, EventArgs e)
{
            if (buffer.Length <= 0) return;

            // Retrieve audio data
            microphone.GetData(buffer);

            double rms = 0; double volume = 0;

            double cMagnitude = 0;
            double cPhase = 0;

            double[] sampleBuffer = new double[
FFT.FourierTransform.NextPowerOfTwo((uint)buffer.Length) ];
            double[] xre = new double[sampleBuffer.Length]; //
Real part
            double[] xim = new double[sampleBuffer.Length]; //
Imaginary part
            double[] spectrum = new double[sampleBuffer.Length /
sampleSize];

            for (int i = 0; i < 2048; i += 2)
            {
                sampleBuffer[index] = Con-
vert.ToDouble(BitConverter.ToInt16((byte[])buffer, i)); index++;
            }

            FFT.FourierTransform.Compute((uint)sampleBuffer.Length,
sampleBuffer, null, xre, xim, false);
            for (int i = 0; i < 512; i++)
            {
                rms += Math.Pow((10.0 *
Math.Log10((float)(Math.Sqrt((xre[i] * xre[i]) + (xim[i] *

xim[i]))))), 2);

                cMagnitude = (float)(Math.Sqrt((xre[i] *
xre[i]) + (xim[i] * xim[i]))); // Magnitude
                cPhase     = (float)(Math.Atan(xim[i] /
xre[i])); // Phase
                spectrum[i] = cMagnitude;
                cMagnitude = 0; cPhase = 0;
            }

            rms /= (double)512;
            volume = (int)Math.Floor(Math.Sqrt(rms));
}
```

At time of writing, in Windows Phone, the sample rate is fixed at 16.000 audio samples per second. According to the Nyquist sample theorem [14], this is appropriate for recording audio up to a frequency of 8 kHz. This is suitable for primitive simple sound sources, like human voice or puffs, but not so efficient with e.g. music.

Each sample is 2 bytes wide and monaural. This implies that each second of recorded sound requires 32 KB, and each minute requires 1.9 MB.

The TimeSpan value is the only option that microphone allows you to specify through the byte size of the buffer passed on GetData(). The BufferDuration property (which is of type TimeSpan) value, must range between 100 ms and 1000 ms (one second) in increments of 10 ms.

Assuming TimeSpan is 100 ms, this means that each time microphone.BufferReady is called we receive a buffer of size 3200 bytes/1600 audio samples.

FFT functions with a number of samples that is a power of two. Applying the FFT to the first 1024 samples means that the frequency resolution

$$\Delta \mathrm{f} = 16000 / 1024 = 15.625$$

The magnitude of each FFT resulting element is referred to the frequencies

$$\Delta \mathrm{f},\ 2\Delta \mathrm{f},\ 3\Delta \mathrm{f}, \ldots,\ (\mathrm{k}/2 - 1)\Delta \mathrm{f}\ (\text{or fs}/2)$$

The most trivial form of sound pattern matching we might use would be to compare the two spectrum magnitudes, calculating the MSE. If the values differentiate within a determined threshold, the puff is detected. Unfortunately this simple approach may not be particularly effective – making puffs using more or less volume can affect the frequency components enough to fail the comparison.

This work uses a more advanced approach, named MFCC (Mel-frequency cepstral coefficients [15]) instead. This method extracts and compares key features in sounds. This approach is more efficient than the simple spectrum comparison - and it is less influenced by ambient noise or the volume of the tone.

The tutorial on [13] provide more in-depth information about FFT on Windows Phone devices.

3.3 Keyboard Layout

The keyboard layout on [7] helped us identify one recurrent issue during the experiments.

Figures 1 and 2 shows the user interface of the layout from [7], where that the keys are arranged linearly and in alphabetical order. This means that if the cursor is standing in front of e.g. “F” and the user wants to select the letter “E”, he would have to wait until the cursor moves through the whole keyboard until he has another chance to select letter “E”.

The work in [11] provides a grid layout of keyboard. The cursor is moved in a way similar to operating a claw crane machine with two buttons. To move the cursor to the target position the user needs to produce two puffs, the first selects the column, and the

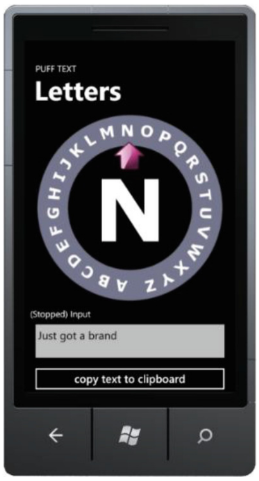

Fig. 1. Screenshot of the first version of PuffText, with a linear round keyboard.

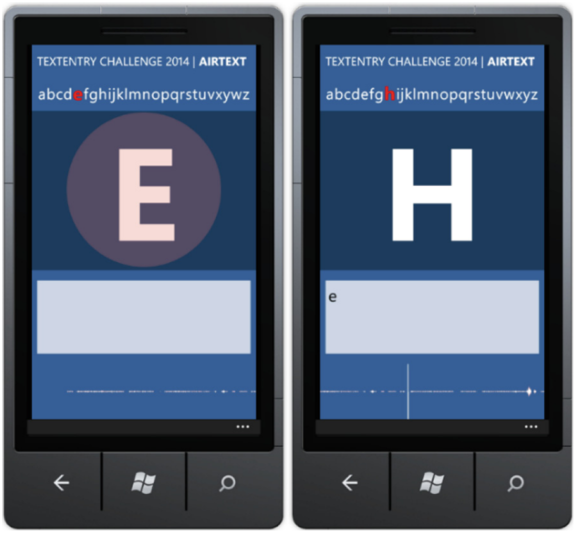
TEXTENTRY CHALLENGE 2014 | AIRTEXT
abcdefghijklmnopqrstuvxywz
E
TEXTENTRY CHALLENGE 2014 | AIRTEXT
abcdefghijklmnopqrstuvwxyz
H
e

Fig. 2. Later version of PuffText, showing a graphical representation of audio volume.

second selects the row. We have reformed this layout and cursor movement pattern approach to the present work (Fig. 3).

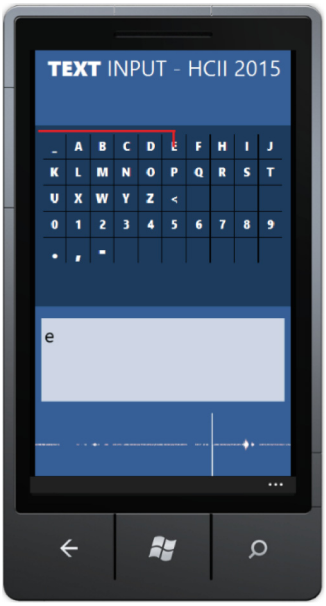

Fig. 3. Grid layout and column-row selection of letters.

4 Experiments

For this round of experiments and interviews we have recruited the some of the same subjects as [7]. On the beginning of this round, we instantly noticed significant improvement in the subject's performance, due to training. Drafting repeated subjects also enabled us to reduce learning session's time and invest more time on training sessions with the new interface.

As soon as we reached the current implementation, 4 rounds of tests were conducted.

Each round consisted in one trying to perform the task of inputting a given text of 47 words.

Some text entry features, such as text prediction are still being tested, for preliminary evaluation simplification. Future works may include a more in depth analysis of the first 2 rounds were meant to be purely instructional. After that, 6 rounds of entering text using the present solution were timed. Three able-bodied and five disabled, high school students of 15 ~ 18 years old, were tested.

The disabled group represented two types of disabilities: having no arms (birth defects) or lack of motor dexterity (due to accidents). The three subjects in this group are unable to handgrip a mobile phone manually.

The users were instructed to keep the distance of approximately 20, 30 and 50 cm during the first three rounds of tests.

These rounds of tests were meant to evaluate, emphatically:

- Puff accuracy at distance

- Background noise immunity – testing ambient noise handling with FFT algorithm
- Shortness of breath due to application usage
- Learnability of column-row letter selection interaction
- Memorability of column-row letter selection interaction
- Speed
- Typing mistakes

5 Results and Discussion

A proof-of-concept application was implemented to perform tests in order to support the argument of this work. The mobile application is able to deliver a way for users to perform text input with a hands-free, speech-free manner. This approach is software based, meaning it will not require extra hardware to function. It also presents itself as a rather discreet and private application, as it implies minimum representation (just puffs) of whatever is being inputted, to the periphery auditory.

The final fastest typing rate registered with this study group (able-bodied and disabled) and the given text was 8.8 words per minute and the average was 7.2. Compare to [6] were a Morse code typist makes 12.6 and a Mouth stick (hardware based) input solution gives 7.88 words per minute. The WPM was measured through logging character/timestamp in a text file on the phone and performing post-test calculation.

The distance of 20 and 30 cm from the user to the phone represented very little difference, although it was observed that the users tended to puff harder when the phone was moved slightly back (10 cm). Even though no shortness of breath episode was noted for these distances, an uninstructed user might report feeling winded. Future works may include a more in-depth analysis of shortness of breath due to puff controlled interfaces.

At the distance of 50 cm we were able to note various unsuccessful attempts, false positives due to background noise and minor shortness of breath. Users described feeling “far from the phone, even for reading”.

Background noise handling algorithm adapts sound level peak threshold and microphone gain to reduce false positives and increase overall performance. The puffs – from a controlled distance of < 25 cm, were detected to a nearly zero false positive performance.

References

1. Arroyo-Palacios, J., Romano, D.M., Exploring the use of a respiratory-computer interface for game interaction. Paper presented at the IEEE Consumer Electronics Society Games Innovation ICE-GIC 2009, London, pp. 154–159 (2009)
2. Sibert, L.E., Jacob, R.J.K.: Evaluation of eye gaze interaction. In: Proceedings of CHI 2000 Conference on Human Factors in Computing Systems. ACM Press, The Hague, pp. 281–288 (2000)

3. Patel, S., Abowd, G., BLUI: Low-cost localized blowable user interfaces. In: Proceedings of the 20th Annual ACM Symposium (2007)
4. Jones, M., Grogg, K., Anschutz, J., Fierman, R.: A sip-and-puff wireless remote control for the apple iPod. Assist. Technol. Off. J. RESNA **20**(2), 107–110 (2008)
5. Sporka, A.J., Kurniawan, S.H., Slavík, P.: Whistling user interface (U3I). In: The 8th ERCIM International Workshop "User Interfaces For All", Vienna, Austria (2004)
6. Levine, S., Gauger, J., Bowers, L., Khan, K.: Comparison of mouth stick and morse code text inputs. Augment. Altern. Commun. **2**(2), 51–55 (1986)
7. Filho, J.F, Valle, T., Prata, W.: Explorations on breathing based text input for mobile devices. In: Proceedings of the 16th International ACM SIGACCESS Conference on Computers and Accessibility, pp. 345–346, ACM (2014)
8. Majaranta, P., Räihä, K.J.: Text entry by gaze: Utilizing eye-tracking. In: MacKenzie, I.S., TanakaIshii, K. (eds.) Text Entry Systems: Mobility Accessibility, Universality. Morgan Kaufmann, San Francisco (2007)
9. Stéphane, N., Lobo, F.G.: A virtual logo keyboard for people with motor disabilities. ACM SIGCSE Bull. **39**(3), 111–115 (2007)
10. Filho, J.F., Prata, W., Valle, T.: Breath mobile: a low-cost software-based breathing controlled mobile phone interface. In: Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services companion, pp. 157–160, ACM (2012)
11. Sporka, A.J.: Text entry using single-channel analog puff input. In: Proceedings of the 16th International ACM SIGACCESS Conference on Computers and Accessibility, pp. 359–360, ACM (2014)
12. Filho, J.F., Prata, W., Valle, T.: PuffText: a voiceless and touchless text entry solution for mobile phones. In: Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility, p. 63, ACM (2013)
13. Galazzo, S.: Sound pattern matching using Fast Fourier Transform in Windows Phone, (2013). http://developer.nokia.com/community/wiki/Sound_pattern_matching_using_Fast_Fourier_Transform_in_Windows_Phone
14. Nyquist-Shannon Sampling Theorem. http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem
15. Mel-frequency Cepstrum. http://en.wikipedia.org/wiki/Mel-frequency_cepstrum
16. American Speech-Hearing Association. http://www.asha.org/public/hearing/Noise/
17. Sakamoto, D., Takanori, K., Takeo, I.: Voice augmented manipulation: using paralinguistic information to manipulate mobile devices. In: Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services, pp. 69–78, ACM (2013)