BLUI: Low-cost Localized Blowable User Interfaces

Shwetak N. Patel and Gregory D. Abowd
School of Interactive Computing & GVU Center, Georgia Institute of Technology
85 5th Street NW, Atlanta, GA 30332-0760,USA
{shwetak, abowd}@cc.gatech.edu

ABSTRACT
We describe a unique form of hands-free interaction that can be implemented on most commodity computing platforms. Our 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. Our approach relies solely on a single microphone, such as those already embedded in a standard laptop or one placed near a computer monitor, which makes our approach very cost-effective and easy-to-deploy. We show example interaction techniques that leverage this approach.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: Interfaces, interaction techniques, hands-free interaction, blowable user interfaces, localization

INTRODUCTION AND MOTIVATION
Hands-free input techniques provide quick, secondary input options, especially when someone's hands are preoccupied with another task. In addition, hands-free control also offers individuals with limited arm control the ability to interact with a user interface. Various strategies to provide hands-free interaction have emerged in our research community. Typical approaches use sound or voice-based interfaces, which focus on the verbal parts of human speech. While this is reasonable for complicated or command-based tasks, it is not well suited for direct, low-level controls such as scrolling, button pressing, or selection. Other approaches use non-speech audio for continuous, low-level control, such as humming or whistling [2, 3, 6]. However, verbal and non-verbal sounds still do not necessarily have an intuitive spatial mapping for direct selection tasks, and the stigma associated with producing loud sounds in public places can reduce the adoption of these technologies. Other interfaces use head or gaze tracking to infer one's intent, but these require additional, sometimes costly, instrumentation and may be hard to control [8].

We describe a unique form of hands-free interaction, called BLUI (Blowable and Localized User Interaction), that can be installed on most commodity computing platforms. BLUI supports blowing at a laptop or computer screen to directly control specific parts of an interactive application, such as blowing at a button to activate it. Physically blowing at a laptop or computer screen creates generic UI events at specific places on the screen. These directed events, similar to mouse events, can then directly control certain interactive parts of an application (see Figures 1 and 2). Localization estimates are produced in real-time. The novelty of our approach is that it relies solely on a single microphone that comes embedded on many laptops, which makes it very cost-effective and easy-to-deploy. In addition, users can be discreet when blowing, because our approach does not rely on the sound but the wind generated when blowing.

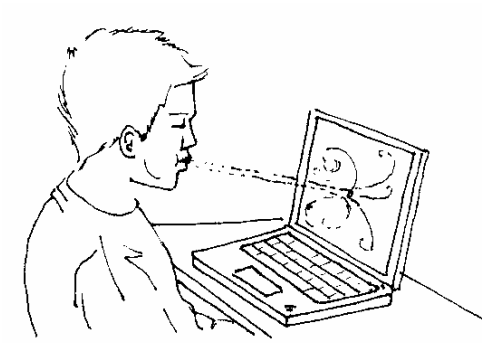

Figure 1: BLUI localizes where someone is blowing on the screen.

Our input method has implications for both hands-free assistive technology applications and entertainment applications that want to leverage the physical blowing metaphor. In this paper, we introduce interactive techniques that leverage our approach. We also discuss the implementation of this system and a preliminary performance evaluation that characterizes accuracy and precision of the localization.

RELATED WORK
Traditional sound-based interfaces use speech recognition to carry out certain command-based actions, such as typing text or controlling an application [4]. Others have looked at using non-verbal sounds to control discrete and continuous low-level events, such as scrolling a window or controlling the movement of a character on the screen [2, 3, 6]. The limitation of these techniques is that they do not provide a direct selection mapping between the sound source and the interface portion of interest (*i.e.,* the user must map a sound to the physical manipulation). Other hands-free approaches

UIST'07, October 7–10, 2007, Newport, Rhode Island, USA.

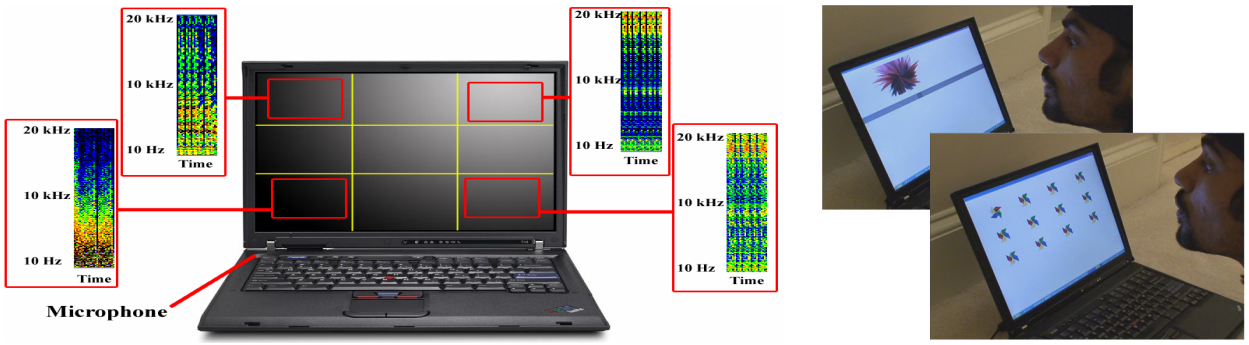

Figure 2: Left: The frequency response at certain regions on the screen. Right: Scrolling and button selection techniques using BLUI.

that use gaze detection, eye tracking, or head tracking to localize where someone is looking at the screen offer more direct interaction [8]. Although these solutions are appropriate for certain applications, they are costly and require additional hardware to be installed. In addition, gaze detection has to be coupled with another input method to indicate a toggle event, such as blinking, and may be harder to control because of involuntary movements.

Various blowing input technologies have emerged from the assistive technology and gaming research communities that use blowing as input to the user interface. The "sip and puff" technique is a popular approach for allowing quadriplegics to control certain interfaces. This technique is limited to only linear controls such as the velocity of a wheelchair or the binary events such as pressing a button or a key. In the entertainment community, some of the latest mobile gaming platforms, such as the Nintendo® DS™, use blowing as input for certain games [5]. Examples include blowing into the microphone to move a sailboat or blowing bubbles. Although similar in spirit to our work, these systems do not try to localize where someone is blowing on the screen. They sense the presence and amplitude of the loud sound source generated by blowing on the microphone.

Researchers have also explored a variety of sound source localization techniques [1, 7]. These approaches typically involve triangulating the source of the loudest sound by using multiple microphones. Although this would provide similar and potentially higher resolution localization than BLUI, our aim is to provide an inexpensive software solution without the need for additional equipment.

BLUI AND LOCALIZED BLOWING APPLICATIONS

BLUI is an event-based system. The BLUI engine recognizes blowing activity from the physical surface and generates blowing events. Application programmers can register callbacks for these blowing events and can create special-purpose widgets that respond appropriately. The blowing event identifies the region on the screen at which it was aimed. The pre-defined regions are equally spaced on the screen and range from a 2x2 grid to a 6x6 grid, depending on the desired resolution and accuracy. A UI programmer can think of the blow event as similar to a mouse event, but at a much coarser spatial resolution. We present some sample hands-free interaction techniques we have built using our BLUI system (see Figure 2). Despite the coarse resolution, these interaction techniques have interesting applications for not only gaming and entertainment applications, but also for multimodal desktop interaction and alternate input for assistive technology applications.

Selection

Blowing can provide a hands-free way to directly select objects on an interface. The selection task consists of two parts: targeting followed by confirmation. Our selection widget consists of visual feedback to indicate that a particular item is being targeted. The targeting feedback is essential; although a user may know the general area where she is blowing, there might be other widgets nearby that could also be possible candidates. Our current implementation uses a pinwheel on the widget that begins to spin or text that animates when the user is blowing at the widget to indicate the inferred selection. After targeting the widget, the user then confirms the selection by quickly blowing harder for about one second, although simple dwell time can also be used to confirm. Further empirical testing can provide additional insights on dwell times for selection.

One issue with selection by blowing has to do with error correction. As we will show below, our current implementation on BLUI is coarse-grained. Selection among a number of tightly packed widgets can be error-prone. Blowing events can be used to nudge a selection tool to adjacent neighbors. This interaction mechanism increases the effective resolution of selection without requiring an improvement in the BLUI localization algorithm. In addition, care must be taken with widget placement, because the size of the selection region can be large depending on grid size. Thus, interactive elements are best displayed as a series of screens instead of placing them all on a single window.

Scrolling

The blow-based scrollbar allows users to scroll a window without having to move their hands away from the keyboard. Jump scrolling is supported by treating the scroll direction indicators as buttons. The user simply blows toward the top or the bottom of the scroll bar (for vertical bars) to cause the page to move in the desired direction.

Dragging

We also support smooth, continuous scrolling, which operates by continuously blowing on the bar and aiming the blowing towards the intended scroll directions. This is eas-

ily implemented as a series of buttons along the scroll bar. This notion of tracking-based blow events can also be extended to other interaction techniques, such as panning a document or gross cursor movements.

Physical Blow Metaphor

The physical action of blowing on the computer screen can be used as a direct input for gaming or entertainment applications that use blowing to richly convey a physical phenomenon. For example, we can produce the effect of blowing out birthday candles, blowing away dust, or playing a variant of whack-a-mole. Blowing also offers interesting interaction with 3D widgets, such as pushing a button back into the screen when blowing at it.

SYSTEM DETAILS AND IMPLEMENTAION

BLUI uses real-time audio analysis and fingerprinting on the incoming microphone data. Our analysis looks at the "wind" noise detected by the microphone rather than speech sound patterns. Wind noise produces a broadband frequency response through the microphone, which is very high in amplitude for low frequency components. After having isolated this response, we infer the location of the blow event based on learned audio fingerprints of various regions of the screen using a machine learning classifier.

Theory of Operation

One can think of a microphone as a pressure sensor that is tuned to sound waves. However, they are also highly sensitive to wind pressure, which is why certain microphones contain windscreens. Wind, such as that created by blowing, is a complex waveform that causes a broadband response through a microphone, even in the presence of a windscreen. This is partly from the actual wind pressure reaching the sensing element and partly from the turbulence created by the interaction between the physical structure of the microphone and the wind.

Our approach leverages the broadband amplitude and phase responses produced by different wind patterns reaching a single microphone. The destructive interference caused by the reflection of waves can produce very different acoustic responses by the microphone. In our case, when someone blows towards the screen, the screen reflects and diffracts the wind before it reaches the microphone. Diffraction causes the waves to be filtered, as characterized by a transfer function $H(f)$.

Various other factors, such as the distance and angle of reflection, also contribute to this function. The transfer function $H(f)$ is also dispersive, *i.e.*, the phase shift is not a linear function of the frequency. In addition, depending on the reflection path, filtering at certain frequencies is also evident. To illustrate this phenomenon, Figure 2 shows a spectrograph of the broadband frequency response when blowing at various parts of the screen. Of note is the fact that there is not only an overall change in amplitude, but also an increase at certain frequencies despite being further away from the microphone. Each region on the screen has a very distinct frequency response signature. Thus, this spatial differentiability makes it possible to use acoustic fingerprinting and classification to infer the location of blow events. We use the amplitude and Fast Fourier Transform (FFT) phase shifts for the fingerprints.

One important consideration is the placement of the microphone. An effective position is near the screen and pointed towards the center location of the screen. Fortunately, embedded microphones in the latest notebook computers are on the top of the keyboard because that location also tends to be the ideal position for speech capture. For desktops, we found that placing the microphone 10 cm from the lower corner of the display screen and pointed towards the center of the screen achieves the desired effect.

Implementation

The software component of our prototype consists of a C++ application that samples the microphone interface and performs an FFT on the incoming signal to separate the component frequencies for our analysis in real-time. The application also produces a waterfall plot, a commonly used frequency domain visualization used for visual inspection (*e.g.,* Figure 2). A second application, written in Java, performs the machine learning and provides the user interface development for the system. The Java application connects via a TCP connection to the FFT application and reads the data values.

Detection and Fingerprint Construction

Our algorithm samples the microphone and performs an FFT on the signal using non-overlapping hanning windows sampled at 44 kHz. Each sample consists of frequency components and its associated amplitude values. From the FFT, we can also compute the relative phase shift. After having isolated the sample, we then create a vector consisting of amplitude and phase shift values for frequency intervals ranging from 10 Hz – 20 kHz. Each vector consists of 1024 16-bit amplitudes value and 1024 corresponding 8-bit phase values. Thus, the 2048 components represent the feature vector and, in turn, represent the fingerprint for that sample. We used principal component analysis (PCA) during the training phase to substantially narrow down the feature space to reduce the classification times.

Classification and Training

In our current approach, we create and label discrete regions of equal size on the desktop and use k-Nearest Neighbor (KNN) classification for inferring which of those regions is targeted. Given a query point, KNN works by finding k examples that are closest in distance to the query point. For KNN classification problems, a majority vote determines the query point's class. In our case, each region is treated as a class and each fingerprint vector serves as the sample point. For our distance measure, we use a simple Euclidean distance.

The training period consists of repeated blow events at given regions on the screen. Three to five seconds of blowing at each region is more than sufficient to gather enough training sample points for that region. Approximately 500 signatures are gathered for the training set for each region. We then use PCA on this gathered training set to narrow

down the feature space. Finally, we use cross-validation among the entire training set to determine the suitable *k* value for the KNN classifier.

For accurate calibration, the user must train the system at approximately the same distance away from the display as he would be during use. In the case of a laptop, the tilt of the laptop screen should also be consistent. In practice, we have found that the laptop screen may be tilted approximately 10 degrees in either direction without significant classification errors. However, this may not be a major problem for fixed monitors connected to a desktop or a situation in which an individual with motor disabilities interacts with a mounted device, for example, on a wheelchair. We can consider multiple "profiles" that involve training the system for different individuals and different physical configurations. Sufficient distance is also needed between the user and the screen. For example, on a 14-inch laptop screen, the effective distance is about 10-20 cm from the center of the screen. To increase the accuracy of the classifier, the user must pivot his head from the centerline of the screen when blowing at different regions, rather that translating relative to the screen. Finally, the general blowing pressure must be kept consistent for accurately classifying the same regions later. In addition, 1500 ms of blowing is required to register an event. Thus, the feedback from the interface proves valuable to help regulate the blowing pressure for optimal accuracy.

PERFORMANCE

We conducted a preliminary evaluation of the accuracy of the BLUI localizer with three different individuals. For each person, our setup consisted of a training/calibration period followed by a set of 25-50 blows (depending on the resolution) toward various regions on the screen. We conducted this test two different times. To accurately determine the ground truth and maintain consistency, individuals wore a head-mounted laser pointer to visually indicate where the person is pointed at the display. In practice, this is not necessary as long as the person using the interface it is the one who trained it.

Table 1: Performance of the BLUI localizer for various resolutions (% of correctly identified regions).

	9 (3 X 3)	16 (4 X 4)	25 (5 X 5)	36 (6 X 6)
Laptop	100%	96%	80%	62%
Desktop	100%	92%	82%	66%

We report the overall number of correctly classified regions at varying resolutions for a laptop and desktop (see Table 1). The regions were of the same size and uniformly distributed in a grid pattern across the screen. We found that our localization approach is very accurate for up to 16 regions and shows promising results for 25 regions. The confusion matrix reveals that most of the misclassifications (84%) were of adjacent regions. Part of the reason for this is because the feature set is related to the spatial arrangement of the regions. The lower accuracies for higher resolution regions are also the result of the un-collimated or conical nature of blowing, because of the possibility of reflection off multiple regions. Moving closer to the screen can correct this issue.

IMPROVEMENTS AND FUTURE WORK

Though we saw promising results with our user interface, there are some important considerations to improve upon in our current design. Although we did not apply background noise filtering, this would be necessary for outdoor and noisy environments, unless a practical sound baffle could be produced that insulates the microphone from ambient noise. We can avoid false positive responses by employing a sophisticated audio filtering scheme that differentiates between the learned broadband wind and other noises.

We used a real-time classification approach to identify discrete regions. An analytical approach that directly models the transfer functions can provide a more continuous input analysis and higher resolution. We also presented some initial performance data of our localization scheme, but an important next step is to conduct empirical user studies to answer interaction questions, such as the selection times for various blow-based interfaces.

CONCLUSION

We presented a system, called BLUI, that enables blowing at a laptop or computer screen to directly control interactive applications. BLUI produces coarse-grained localization estimates in real-time to determine where on the screen the person is blowing. Because our approach is does not require any additional hardware or instrumentation, it is also cost effective. Results also show we can localize up to 16 regions on a laptop or desktop with over 95% accuracy.

REFERENCES

1. Bian, X., Abowd, G.D., and Rehg, J.M. Using Sound Source Localization in a Home Environment. In Pervasive 2005. 2005. pp. 19-36.
2. Harada, S. et al. The Vocal Joystick: Evaluation of Voice-based Cursor Control Techniques. In ASSETS 2006. 2006. pp. 189-196.
3. Igarashi, T. and Hughes, J.F. Voice as sound: Using nonverbal voice input for interactive control. In the Proc. of UIST 2001. 2001. pp. 155–156.
4. Karimullah, A.S. and Sears, A. Speech-based cursor control. In SIGCRAPH 2002. 2002. pp. 178–185.
5. Nintendo DS. http://www.nintendo.com/ds 2007.
6. Olwal, A. and Feiner S. Using Prosodic Features of Speech and Audio Localization in Graphical User Interfaces. In IUI 2005. 2005. pp. 284-286.
7. Scott, J., Dragovic, B. Audio Location:Accurate Low-Cost Location Sensing. In Pervasive 2000. pp.1-18.
8. Zhu, Z and Ji, Q. Eye and Gaze Tracking for Interactive Graphic Display. In Machine Vision and Application. Volume 15. Number 3. July 2004. pp. 139-148.