SoundWave: Using the Doppler Effect to Sense Gestures

Sidhant Gupta[1,2], Dan Morris[1], Shwetak N Patel[1,2], Desney Tan[1]

[1]Microsoft Research
Redmond, WA, USA
{dan, desney}@microsoft.com

[3]University of Washington,UbiComp Lab
Seattle, WA, USA
{sidhant, shwetak}@uw.edu

ABSTRACT

Gesture is becoming an increasingly popular means of interacting with computers. However, it is still relatively costly to deploy robust gesture recognition sensors in existing mobile platforms. We present *SoundWave*, a technique that leverages the speaker and microphone already embedded in most commodity devices to sense in-air gestures around the device. To do this, we generate an inaudible tone, which gets frequency-shifted when it reflects off moving objects like the hand. We measure this shift with the microphone to infer various gestures. In this note, we describe the phenomena and detection algorithm, demonstrate a variety of gestures, and present an informal evaluation on the robustness of this approach across different devices and people.

Author Keywords

In-air gesture sensing; Doppler; interaction technique.

ACM Classification Keywords

H.5.m Information interfaces and presentation: Miscellaneous

General Terms

Human Factors, Design.

INTRODUCTION AND MOTIVATION

Recent advances in computer vision techniques have popularized hand and body gestures for interacting with computers. For example, the Toshiba Qosmio G55 laptop uses its front-facing RGB webcam to allow the user to control PowerPoint slides or music/video playback. Unfortunately, vision-based gesture recognition techniques are generally brittle (e.g., sensitive to lighting conditions) and require quite a bit of processing power. The Microsoft Xbox Kinect is another example of a successfully deployed computer vision system, but miniaturizing this technology and making it practical for mobile devices may take some time.

As an alternative, sonic gesture sensing has been shown to be a reliable tool for sensing a variety of in-air gestures for controlling interfaces. Current technologies, however, have focused on separate transducers and receivers rather than leveraging arguably the most ubiquitous components in computing systems: the speaker and microphone.

CHI'12, May 5–10, 2012, Austin, Texas, USA.

Figure 1: SoundWave allows non-contact, real time in-air gesture sensing on existing commodity computing devices.

To this end, we present SoundWave, a sound-based gesture sensing approach that utilizes the existing audio hardware of mobile devices. This technique uses a well-understood phenomenon known as the "Doppler effect" or "Doppler shift", which characterizes the frequency change of a sound wave as a listener moves toward or away from the source. A common example is the change in pitch of a vehicle siren as it approaches, passes, and then moves away from the listener. Using this effect, SoundWave detects motion in front of and around a computing device and uses properties of the detected motion – such as speed, direction, and amplitude – to recognize a rich set of gestures. For instance, the direction and speed of a hand moving up or down can be sensed to scroll a webpage in real-time (see sketch in Figure 1 as well as accompanying video figure). SoundWave can also, for example, detect two hands moving in opposite directions, which we use as a "rotation" gesture in our example applications. Unlike vision, SoundWave can detect gestures without line of sight, making it complementary to vision-based systems.

We are not the first to use sonic techniques or the Doppler effect for gesture and motion sensing. For example, Tarzia et al. measure the intensity of the echoes received by a microphone to detect human presence and attention [5]. Paradiso et al. made use of a continuous 2.4 GHz tone to drive custom patch antennas. They used the reflected Doppler-shifted signal to infer human motion and upper body kinematics in an interactive space [3]. More recently, Kalgaonkar et al. developed a device to recognize one-handed gestures in 3D space using low-cost ultrasonic transducers that emit a 40 kHz tone. They placed one transmitter and two receivers in a triangle pattern where gestures could be per-

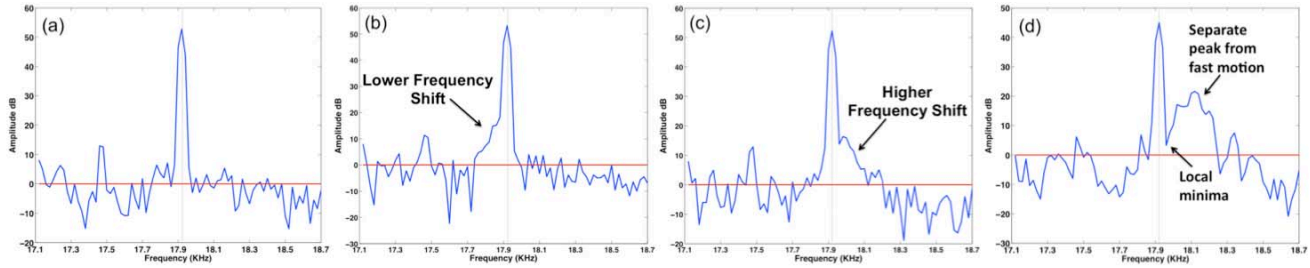

Figure 2: (a) Pilot tone with no motion. (b and c) Increase in bandwidth on left and right due to motion away from and towards the laptop respectively. (d) Shift in frequency large enough for a separate peak. A single scan would not capture the true shift in frequency and would terminate at the local minima. A second scan compensates for the bandwidth of the shifted peak.

formed and sensed [4]. While these projects show the potential of low-cost sonic gesture sensing, they require custom hardware, which is a significant barrier to widespread adoption. In our work, we focus on a solution that works across a wide range of existing hardware to facilitate immediate application development and adoption.

THE SOUNDWAVE SYSTEM

SoundWave uses existing speakers on commodity devices to generate tones between 18-22 kHz, which are inaudible. We then use the existing microphones on these same devices to pick up the reflected signal and estimate motion and gesture through the observed frequency shifts.

Theory of Operation

The phenomenon SoundWave uses to sense motion is the shift in frequency of a sound wave in response to a moving object, an effect called the Doppler effect. This frequency shift is proportional to source frequency and to the velocity with which the object moves. In our approach, the original source (the speakers) and listener (the microphone) are stationary, thus in absence of any motion, there is no frequency change. When a user moves his hand, however, it reflects the waves, causing a shift in frequency. This frequency is measured by the microphone ($\mathbf{f_r}$) and can be described by the following equation, which is used for Doppler radar as well as for estimating frequency changes in reflection of light by a moving mirror [2]:

$$f_r = f_t \cdot \left(\frac{c + v}{c - v}\right)$$

$where, f_r = perceived\ frequency\ at\ microphone;$
$f_t = original\ frequency\ from\ speaker;$
$c = speed\ of\ sound\ in\ air;$
$v = velocity\ of\ target/hand$

Figure 2 shows the frequency of the signal (a) when no motion is present and when a hand is moved (b) away from or (c) closer to the laptop. This change in frequency as a hand moves farther or closer is one of the many characteristic properties of the received signal that we leverage in detecting motion and constructing gestures.

Algorithm & Implementation Details

SoundWave generates a continuous pilot tone, played through the device's speakers at the highest possible frequency (typically in the range of 18-22 kHz on commodity audio systems). Although we have verified that SoundWave can operate on audio down to 6 kHz, we favor tones above 18 kHz since they are generally inaudible [1]. Additionally, the higher the frequency, the greater the shift for a given velocity, which makes it computationally easier to estimate motion at a given resolution. The upper bound is largely a function of most laptop and phone speaker systems only being capable of producing audio at up to 22 kHz. Fortunately, we do not need much higher frequencies to sense the relatively coarse gestures we are targeting.

Due to variations in hardware as well as filtering in sound and microphone systems, SoundWave requires an initial calibration to find the optimal tone frequency (no user intervention is required). It performs a 500 ms frequency sweep, and keeps track of peak amplitude measurements as well as the number of candidate motion events detected (i.e., potential false positives). SoundWave selects the highest frequency at which minimum false events are detected and the peak is most isolated (i.e., the amplitude is at least 3 dB greater than next-highest peak in the sweep range). The system consistently favors the 18-19 kHz range.

With the high-frequency tone being emitted, any motion in proximity (around 1 m depending on speed) of the laptop will cause Doppler-shifted reflections to be picked up by the microphone, which is continuously sampled at 44.1 kHz. We buffer the incoming time-domain signal from the microphone and compute the Fast Fourier Transform (FFT) with 2048-point Hamming window vectors. This yields 1024-point magnitude vectors that are spread equally over the spectral width of 22.05 kHz. After each FFT vector is computed, it is further processed by our pipeline: signal conditioning, bandwidth extraction, motion detection, and feature extraction.

Signal Conditioning: Informal tests with multiple people indicated that the fastest speed at which they could move their hands in front of a laptop was about 3.9 m/sec. Hence, we conservatively bound signals of interest at 6 m/sec. Given our sampling rate and FFT size, this yields about 33 frequency bins on either side of the emitted peak.

Bandwidth Extraction: As seen in Figure 2, motion around the device creates a shifted frequency that effectively increases the bandwidth of the pilot tone (i.e., window averaging and spectral leakage blur the movement of the peak). To detect this, SoundWave computes the bandwidth of the pilot tone by scanning the frequency bins on both sides in-

dependently until the amplitude drops below 10% of the pilot tone peak. Using a relative amplitude drop allows the system to respond dynamically, such as when the user changes the volume of the speakers.

For most cases (e.g. Figure 2b and 2c), this is sufficient for inferring motion. However, if the shift is large enough, the reflected signal separates from the pilot tone's peak rather than blurring the peak (e.g. Figure 2d). To address this, we perform a second scan, looking beyond the stopping point of the first scan. If a second peak with at least 30% of the primary tone's energy is found, the first scan is repeated to find amplitude drops calculated from the second peak.

To verify our approach, we analyzed various hand motions at different speeds. Using our percentage-based thresholds, we found that motion can be detected in each case with near-perfect accuracy. We note that we did not change these percentage thresholds as we tested SoundWave on different computing devices or with different people.

Motion Detection and Feature Extraction: The frequency vectors have a per-bin resolution of 21.5 Hz. With a pilot tone of 20 kHz this translates to detecting movements as slow as 18.5 cm/sec. In practice, we have found that the bandwidth of the pilot tone itself with no motion is ~80 Hz, which can vary from 60-120 Hz (1-3 bins on either side of the tone) depending on the quality of the sound system. Thus, we consider a "motion event" to occur when there is a frequency shift bandwidth of 4 or more bins. We have found that this threshold allows sufficiently slow movements of the hand to be detected while ignoring false positives due to variations in the bandwidth.

Measurable Properties

In addition to the fundamental frequency shift, we can also compute other useful features for inferring gestures.

Velocity: The measured frequency change is proportional to the absolute speed of the target. SoundWave can measure the difference between the original and reflected frequencies to differentiate slow, medium, and fast gestures.

Direction: Determining whether the hand is moving toward or away from the computing device can be made from the sign of the frequency shift. A positive shift indicates movement toward the device.

Proximity and Size of Target: The amplitude of the observed signal increases as the target moves closer to the computing device, and it also increases with size and reflectivity of the target. For example, a larger hand or open palm manifests as larger amplitude than a smaller or fisted hand.

Time Variation: Measuring the variation of the above three properties over time allows us to both observe the rate of change and use it for filtering spurious signals. For example, any motion that lasts for a very short period of time can be reliably filtered out, while longer lasting motion events can be used to identify activities like walking toward or away from the device.

GESTURES AND USE CASES

The features described above can then be combined to form complex gestures (see Video Figure for demonstration).

Scrolling: We found that mapping motion events directly to control scrolling, such as for a web browser, works quite well. However, a clutching mechanism is required to prevent inadvertent scrolling as the hand returns to a particular position. Using the velocity feature and scrolling only when it meets a certain speed criterion makes this possible. We also investigated using a 'double-tap' gesture to activate scrolling and using an idle timeout for deactivation.

Single-Tap or Double-Tap: By observing the change in direction over time, the 'frequency' at which the direction is changing can be computed. The value of this direction frequency can be used for detecting tap gestures, which can be further used to distinguish quick taps, much like a mouse double-click, from slower taps. In a Tetris application, we mapped slow taps to 'left' and quick taps to 'right' and were able to maneuver with reasonable precision.

Two-Handed Seesaw: This gesture requires moving both hands simultaneously in opposite directions at the same time. It is detected by the presence of both up- and down-shifted frequency components in the same FFT vector. We mapped this gesture to rotation action in the Tetris game.

Sustained Motion: This gesture is activated when at least N consecutive motion events in the same direction are detected. A large N can signify that a person is walking (as opposed to moving the hand, which has fewer consecutive motions). We have used the walking gesture to automatically put a computer to sleep or wake it up as a user walks away from or toward it. In a game of Tetris, we mapped a 'pull back' gesture (sustained motion of N=10 events away from the device) to the 'drop block' action. Lastly, we implemented a 'flick' gesture with a sustained-motion threshold of N=5 events to allow users to browse a photo album by moving her hands left or right; in this case we also put a maximum and minimum limit on gesture speed.

PRELIMINARY PERFORMANCE EVALUATION

We performed a set of preliminary tests to evaluate how well SoundWave works across devices and people and to estimate accuracy and robustness in different environments.

Generalizability Across Laptops and People

To support our claim that SoundWave could potentially work with most commodity computing platforms, we tested SoundWave on 11 different computers: five desktop PCs (4 Dell, 1 HP), 2 MacBook Pros (15" & 13"), a Lenovo T61p, an IBM Thinkpad T43, a Dell Studio 1555, and a HP EliteBook laptop. We found that all of them performed similarly to our performance results *without any changes* to the algorithms or thresholds. This also included two desktop PCs with an external USB soundcard and microphone.

To ensure that SoundWave works across people, we tested it with 6 individuals. We asked them to control 3 applica-

Location	Two Handed	Pull Back	Flick	Quick Taps	Slow Taps
Home	96.67	95.00	98.33	86.67	96.67
Cafe	100	96.67	93.33	88.33	93.33

Table 1: Average % of correctly recognized gestures across three users and two sessions in quiet and noisy locations.

tions using various gestures: (1) scrolling a webpage using simple hand motion toward or away; (2) playing Tetris using two-handed seesaw, pull-back, double-tap, and slow-tap gestures; and (3) browsing pictures in a cover flow layout with a 'flick' gesture. Although it took a few minutes for users to understand how to perform certain gestures, all users were able to successfully control all 3 applications.

Accuracy and Robustness

To measure how well gestures can be detected using SoundWave, we asked 3 users (1 female) aged 25-28 to perform 5 different gestures. Each user performed 10 repetitions of each gesture in both quiet and noisy environments. The first was in a home environment (noise level ~45 dB SPL) and the second in a noisy cafeteria (noise level ~72 dB SPL). This task was repeated twice for each user. In total, 600 gestures were performed. SoundWave performed well irrespective of the location (Table 1). This was especially the case for two-handed gestures. Quick taps performed the worst since users tended to move their fingers rather than their palm; different fingers generated different velocity components. However, this may not be a problem in applications where there is visual feedback.

To measure the number of times any unintended motion was detected, we conducted an hour-long test in each of the two locations. Users sat in front of the laptop, but neither performed any explicit gesture nor typed on the keyboard. For the home environment, an average of 2.5 false motion events occurred per minute, whereas for the café 6 events per minute were detected. Though relatively high, setting a threshold of N=4 for consecutive events eliminates the interpretation of these 'motions' as 'gestures.' Here N means the number of consecutive motion events or FFT frames, i.e. the gesture needs to last for at least 4 frames or ~185ms for it to be even considered for gesture recognition. Therefore, although motion was detected, post-processing these events with N=4 resulted in 0 false gesture detections, i.e. they were classified as 'noise'.

Because laptop microphones are generally housed in the bezel around the keyboard, the number of false events detected greatly increases when a user types. We mitigate this by disabling SoundWave when we know the user is typing, similar to what track-pads do to prevent accidental input.

We also confirmed that we are able to play audible music on the same laptop while successfully detecting motion events. We found that music does not harm performance, because frequencies seldom conflict and the threshold adapts. Additionally, we tested the range of detection by setting the volume of the speakers, which in turn controls how loud the pilot tone is. The volume control can be used to regulate the effective detection range, which could be useful in crowded situations in which the user may explicitly not want to pick up ambient movement.

LIMITATIONS & IMPROVEMENTS

SoundWave is a promising approach for sensing interactive in-air gestures with no additional hardware requirements. However, it is not without limitations. The key drawback of this approach is the dependence on a tone, which may be audible and possibly annoying for children and pets. In addition, some devices incorporate filtering that prevents tone generation or recording over 18 kHz; a potential solution to this problem is "piggy-backing" a tone on a user's digital music. Additionally, using Doppler shift inherently limits detection to motion gestures, thus requiring other complimentary techniques for detection of static poses.

In this work, the algorithms presented were implemented and tested on various laptops and desktop PCs, however this approach extends to smart phones and tablets. Anecdotally, we observe the same frequency shift when performing gestures in front of mobile phones. Computational complexity and power requirements on such devices can be further reduced by using Goertzel's algorithm for computing selective frequency bins instead of a complete FFT.

We believe gesture sets could be extended beyond the ones presented here by using techniques like Hidden Markov Models for multi-state gestures. Many newer mobile devices also have multiple speakers and microphones that we could leverage for gesture localization.

CONCLUSION

In this paper, we described the use of the Doppler effect to build a software-only solution capable of sensing motion gestures on commodity computing hardware. Furthermore, we detailed a robust algorithm for detecting motion events and using characteristics of the sensed signal for implementing two-handed gestures, as well as more complex gestures such as double-tap. Lastly, we showed the robustness of the approach across different devices, users, and environments.

REFERENCES

1. D'Ambrose, C. "Frequency Range of Human Hearing". *The Physics Factbook* 2003. Downloaded 10/1/2011.
2. Ditchburn, R.W. *Light*. Dover publications, p. 331-333. 1961, 1991.
3. Paradiso, J., Abler, C., Hsiao, K. and Reynolds, M. The magic carpet: physical sensing for immersive environments. *In Proc. ACM CHI 1997*.
4. Kalgaonkar, K. and Raj, B. One-handed gesture recognition using ultrasonic Doppler sonar. *In Proc. IEEE Acoustics, Speech and Signal Processing 2009*.
5. Tarzia, S.P, Dick, R.P. Dinda, P.A and Memik, G. Sonar-based measurement of user presence and attention. *In Proc. ACM UbiComp* 2009.