Ubiquitous Keyboard for Small Mobile Devices: Harnessing Multipath Fading for Fine-Grained Keystroke Localization

Junjue Wang, Kaichen Zhao and Xinyu Zhang
University of Wisconsin-Madison
{jwang283, kzhao25}@wisc.edu, xyzhang@ece.wisc.edu

Chunyi Peng
The Ohio State University
chunyi@cse.ohio-state.edu

ABSTRACT

A well-known bottleneck of contemporary mobile devices is the inefficient and error-prone touchscreen keyboard. In this paper, we propose UbiK, an alternative portable text-entry method that allows user to make keystrokes on conventional surfaces, *e.g.*, wood desktop. UbiK enables text-input experience similar to that on a physical keyboard, but it only requires a keyboard outline printed on the surface or a piece of paper atop. The core idea is to leverage the microphone on a mobile device to accurately localize the keystrokes. To achieve fine-grained, centimeter scale granularity, UbiK extracts and optimizes the location-dependent multipath fading features from the audio signals, and takes advantage of the dual-microphone interface to improve signal diversity. We implement UbiK as an Android application. Our experiments demonstrate that UbiK is able to achieve above 95% of localization accuracy. Field trial involving first-time users shows that UbiK can significantly improve text-entry speed over current on-screen keyboards.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Input devices and strategies*

Keywords

UbiK; mobile text-entry; paper keyboard; acoustic localization

1. INTRODUCTION

Despite the increasing sophistication of mobile technology, interacting with today's mobile devices can involve painful contortions. Miniature circuits and displays keep pushing portable devices to a smaller form factor — down to stamp-size for emerging wearable computers — but human fingers and hands do not shrink accordingly. As a result, on-screen keyboard, a daily part of life for many people, remains as an obstacle that prevents the anticipated role switching for mobile devices from information consumers to providers.

This grand challenge has prompted considerable research in the field of mobile text entry. Existing work in human computer interaction addressed the problem by redesigning keyboard layout [1], adapting key size [2], expanding keyboard area [3, 4], *etc.* However, users are highly resistant to learning new methods, particularly new keyboard layouts or key shapes [5]. Projection keyboard [6–8] provides a more palatable solution for heavy typists, but they require bulky new hardware to accompany mobile devices, which compromises their portability.

In this paper, we propose UbiK, a new approach to mobile text input that recognizes keystrokes through fine-grained localization. UbiK enables PC-like text-entry experience, by allowing users to click on solid surfaces, and then localizing the key symbol through the keystroke's acoustic patterns. A keyboard outline can be drawn on the surface, or printed on a piece of paper atop. The keystrokes can be sensed using microphones that are readily available on today's mobile devices. With such simple setup, UbiK can serve as a spontaneous and efficient keyboard in a wide range of scenarios, *e.g.*, on an office desk, conference room table, or an airplane tray table.

The key challenge for UbiK lies in fine granularity. Inter-key distance on typical PC keyboard is only around 2 centimeters. Wireless localization, even those requiring user to carry active radios, can only achieve several feet of granularity [9]. Using microphone arrays, existing sound source localization algorithms [10] can achieve a few meters of accuracy based on time-difference-of-arrival (TDOA) information. Theoretically, sound waves are coherent within their meters-scale wavelength, and audio sources within this range are hardly distinguishable. However, this holds only for point sound source in free-space. Practical clicks on solid surfaces generate entangled audio waves that undergo complex multipath reflection patterns on the surface and the body of the mobile device, as they propagate towards the microphone. Although such patterns worsen the unpredictability and compromise accuracy of audio ranging [11], they can create location-dependent miniature sound signatures, thus improving the granularity of sound source localization.

To verify the above principle, we use commercial off-the-shelf (COTS) smartphones to conduct a comprehensive measurement study of acoustic multipath patterns produced by keystrokes on conventional solid surfaces. We find that finger clicks on the same spot exhibit highly consistent fading patterns, due to soundwave reflections cancelling or strength-

MobiSys'14, June 16 - 19 2014, Bretton Woods, NH, USA

http://dx.doi.org/10.1145/2594368.2594384.

ening each other. Such patterns depend on the sound frequency or wavelength, and can be characterized by the amplitude spectrum density (ASD) of the click sound. A more important observation is that the ASD of different keystroke locations reveals highly distinguishable profiles and can be conveniently used as location signatures. The signatures exhibit a certain level of correlation within a short distance of several millimeters, but the correlation diminishes monotonically as physical separation increases. Since neighboring key distance on a PC keyboard is roughly 20 millimeters, ASD has potential to enable keystroke-level location granularity.

UbiK synthesizes these experimental observations to realize a fine-grained, fingerprinting-based keystroke identification system comprised of three key components: detection, localization and adaptation. We design an online detection algorithm that adapts noise-floor threshold to single out keystroke signals, and augments motion sensors to isolate the impact of bursty interference such as human voices. We introduce a keystroke localization framework that uses simple nearest neighbor search for signature matching, while optimizing the signatures to maximize the feature separation between keys. In particular, we leverage dual-microphone interface on typical mobile devices to improve the audio signal diversity and hence signature diversity. We formulate an optimization-based solution to cap the frequency range of the ASD profile, so as to prevent the noisy features from polluting localization accuracy. In addition, we take advantage of the unique opportunity offered by users' online feedback to calibrate the training signatures and discriminate their significance.

We build UbiK as a prototype application for Android devices. Our implementation achieves real-time keystroke detection and localization without noticeable latency. Our baseline evaluation demonstrates that UbiK can easily achieve 90+% of localization accuracy, even with 3 training instances per key and without user calibration. Coupled with its online adaptation, its accuracy quickly escalates to around 95%. UbiK works consistently on a variety of solid surfaces and keyboard layouts. An experimental study involving first-time users show that UbiK maintains high performance across different users. UbiK is robust against minor displacement of the keyboard or mobile devices, and can rapidly converge to high accuracy after significant disturbance. Typing is not as rapid as on a mechanical keyboard but easily outperforms thumb-operated keyboards.

The main contribution of this paper is to address mobile text entry problem using a fine-grained keystroke localization system. This contribution breaks down into the following aspects:

- We provide measurement based evidence that verifies the feasibility of fine-grained, centimeter scale click sound localization using acoustic multipath signatures.
- We design UbiK, a practical framework that optimizes the location signatures and enables detection/localization of keystrokes on solid surfaces and in real-world environment.
- We implement UbiK as an efficient application running on COTS Android devices, and further validate UbiK's performance through comprehensive micro-benchmark tests and users' field trials.

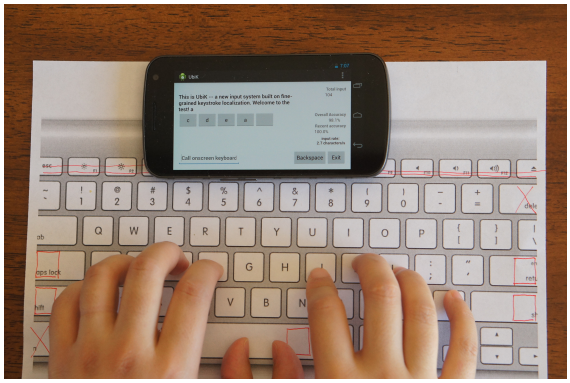


Figure 1: A typical use case of UbiK.

The remainder of this paper is structured as follows. Section 2 presents an overview of the operations, architectures and design objectives underlying UbiK. Section 3 elaborates on our feasibility study of UbiK and verifies the premises behind it. Then, Sections 4, 5 and 6 describe UbiK's main modules in detail. Section 7 presents our implementation of UbiK on Android, followed by a comprehensive experimental evaluation in Section 8. We discuss UbiK's limitations in Section 9 and related work in Section 10. Finally, Section 11 concludes the paper.

2. UbiK OVERVIEW

UbiK facilitates small form-factor, touchscreen-based mobile devices with an *external, virtual, paper-printed*[1] keyboard. Although such a keyboard does not provide the same kinesthetic feedback as a mechanical one, it saves the precious touch-screen area and allows ten-finger typing on a larger workspace. Unlike on-screen virtual keyboard, UbiK is insensitive to gentle taps and touches. It allows finger/wrist rest on the touch surface, thus relieving fatigue [2] caused by hovering.

Figure 1 illustrates a typical use case of UbiK. The mobile device is placed near the printed keyboard, so as to capture the keystroke sound using its microphones. Before running UbiK, an initial setup is needed, whereby the user types all the printed keys at least once and generates "training sounds". UbiK runs a set of novel keystroke detection/localization mechanisms in the mobile device, which learn highly distinguishable acoustic features from the keystrokes, and then use such features to detect subsequent keypress events and localize the corresponding keys.

Usage conditions UbiK is applicable under two basic conditions: *(i)* The surface can generate audible sounds when the user clicks it with fingertip and nail margin. *(ii)* Throughout the usage life-cycle, the positions of the mobile device and the printed keyboard do not change significantly.

The first condition can be easily satisfied in real-life environment. Through experiments, we show that UbiK works on a wide range of surfaces, *e.g.*, on top of a wood table, hard-covered paper, metal cabinet, plastic board, *etc.*. UbiK does not rely on the timbre of the keystroke sounds.

[1]Throughout this paper, we print the keyboard on a letter-sized paper. But any tangible and visible keyboard layout (*e.g.*, drawn on a surface) works for UbiK. The keyboard outline is not mandatory. It mainly serves as a visual assistant that helps users to maintain keystroke positions consistent.

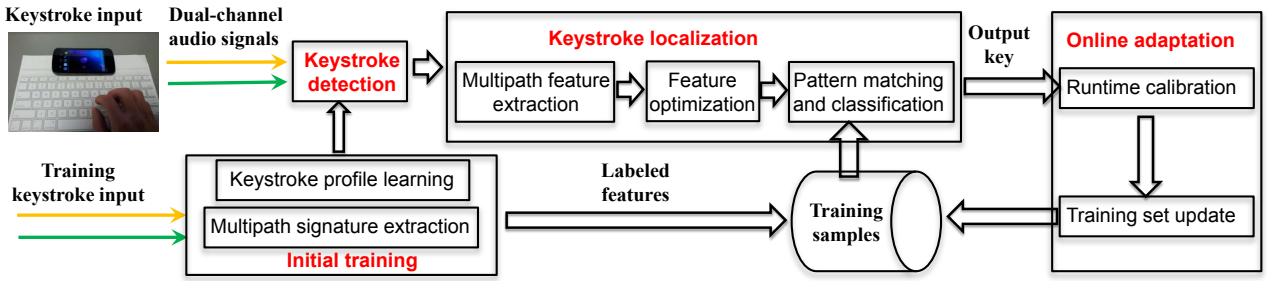


Figure 2: Architecture of UbiK.

Keystrokes can be identified even on a flat, homogeneous surface that makes the same audible sound everywhere.

Regarding the second condition, we note that keyboard input is necessary only when the mobile device acts as a static screen, where user can type and monitor the input text in real-time. Thus, it is reasonable to assume the printed keyboard and mobile devices sit on fixed positions that can be identified using arbitrary anchoring marks on the surface. The user can always move the mobile device away and reposition it back to the anchoring points to continue UbiK's usage life cycle. But whenever the keyboard and mobile device are put to a new location or surface, UbiK requires repeating the initial setup to start a new life cycle.

Design goals and challenges UbiK is designed to meet the following goals, which are geared towards similar user experience as on a desktop keyboard.

(i) Portability. UbiK should not rely on any extra hardware, bulky keyboards or external infrastructure support. It should allow spontaneous setup and usage, using only hardware/software built in existing mobile devices.

(ii) Fine-grained, centimeter-scale keystroke localization. Typical inter-key separation on a PC keyboard is only around 2 cm when printed on letter-sized paper. Thus, UbiK needs a localization mechanism that matches the centimeter-scale granularity and can identify keystrokes with high accuracy. There exists a vast literature of algorithms for audio-source localization leveraging Time-Difference-of-Arrival (TDOA) or energy difference between multiple microphones [10, 12]. Yet such algorithms can only achieve meter-scale accuracy in practical environment with rich reverberation effects.

(iii) Processing efficiency. UbiK's spontaneous keyboard setup requires user to traverse all keys to generate training data for localization. Such training procedure must be brief and should not compromise usability. Ideally, a few repetitions of training should ensure high localization accuracy. In addition, since UbiK runs on the mobile device directly, it must process the keystrokes without any noticeable latency.

(iv) Robustness. UbiK's localization mechanism must be resilient against minor displacement of the keyboard or mobile device, which may be caused by unintended movement or inaccurate repositioning during the usage life cycle. It should not be affected significantly by variations of user's finger/hand posture. In addition, UbiK should accurately detect the presence of keystrokes even in noisy environment.

System architecture UbiK architects the following three major components to build a full-fledged keystroke localization system for mobile devices.

(i) Keystroke detection. UbiK runs a keystroke detection algorithm that takes advantage of the audio signal onset patterns generated by keystrokes. It isolates noise and interference by fusing audio and motion sensing data. It is used to trigger the subsequent keystroke localization.

(ii) Keystroke localization. Instead of attempting to combat multipath reflections as in existing localization schemes [13, 14], UbiK's keystroke localization algorithm harnesses the location-dependent audio signal cancellation/enhancement features, and optimize them to achieve high accuracy at a fine granularity. Further, UbiK migrates the principles of multi-antenna spatial diversity in wireless communications [15], and uses dual-microphones on typical mobile devices to enhance localization accuracy.

(iii) Online calibration and adaptation. UbiK takes advantage of run-time user feedback on-screen to correct occasional localization errors. It further employs an online adaptation algorithm to refresh the training data to prevent error propagation.

Figure 2 illustrates the work-flow inside UbiK. During the initial training stage, UbiK learns acoustic parameters and features that later assist run-time keystroke detection and localization. Afterwards, it runs the keystroke detection algorithm to extract audio signals specifically generated by key presses. The keystroke localization algorithm extracts and optimizes acoustic features from those signals and finds the best match in the trained benchmark. It then outputs the resulting key symbol along with alternative candidates on-screen, which can be calibrated by the user. The calibrated result is fed back to the online adaptation algorithm to refresh the training data.

3. FEASIBILITY STUDY OF UbiK

UbiK is built on the hypothesis that audio channel's multipath profile can enable fine-grained keystroke distinction. In particular, the audio channel contains a rich set of characteristics that is consistent over time for each key, but differs across key locations. In this section, we conduct a comprehensive set of experiments to verify this hypothesis through addressing the following three questions: *(i)* Do different keystrokes generate unique audio signatures? *(ii)* Do these signatures exhibit fine-granularity and temporal stability? *(iii)* Can these signatures be enhanced using COTS hardware?

3.1 Preliminaries

We first describe the preliminaries to the feasibility study, covering the basics of sound signals and our experimental setting.

Basics of Sound Signals Sound is a wave phenomenon in air, fluid or solid medium. Mechanical vibrations at a sound source causes compression and decompression of the

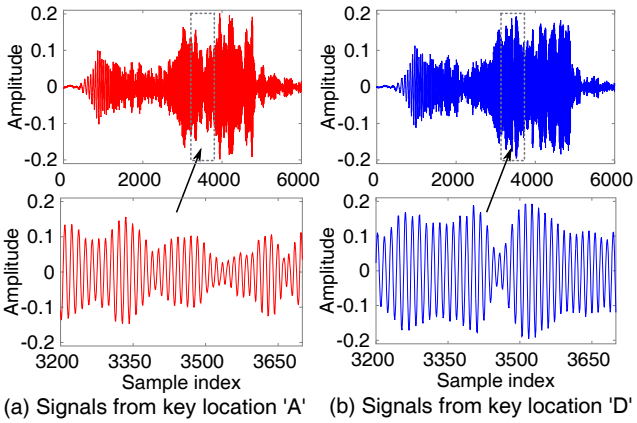


Figure 3: Received chirp signals sent by a speaker at two different key locations: 'A' and 'D'.

medium, which propagates over distance and attenuates following a inverse-square law [16].

Practical audio channel adds more intricacies than attenuation. Solid surface or objects near sound source or microphones can reflect or scatter the original audio source, causing a myriad of "image sources". Phantom waves produced by such image sources can either cancel or strengthen the original wave at different locations. Even for the same location, a sound wave can either be faded or strengthened, depending on its wavelength or frequency. In UbiK, we aim to extract such location and frequency dependent features from keystroke sounds to pinpoint the keystroke location.

Experimental Setup Our experiment setup complies with UbiK's typical usage scenario. We print the Apple Wireless Keyboard (AWK) layout on a piece of letter paper, and put it on a solid surface – a wood table by default. A Galaxy Nexus (GT-I9250) Android smartphone is placed close to the top-level of the printed keyboard to capture the keystroke sounds, at 48 kHz sampling rate. We redrew the exceptionally big keys on AWK, including Shift, Enter, Space, *etc.*, and limit them to be the same size as others. All experiments run in an office environment, with moderate noise coming from desktop computers and a server room nearby. Our experiments require a dual-microphone setup. Though equipped with two microphones, the Android application framework only allows user access to one microphone. We circumvented this limitation by enabling the Tinyalsa driver in Android OS (Section 7).

We next present our experimental validation of the aforementioned hypothesis.

3.2 Multipath Channel Profile-based Signature

Frequency and location dependent fading effects We first design experiments to understand the variation of multipath fading effects across different frequencies and locations. We place a smartphone's speaker at two key locations, 'A' and 'D', on the printed AWK. The speaker emits a 100 ms chirp sound, with magnitude being constant but frequencies linearly increasing from 10 Hz to 5 kHz. Fig. 3 plots the audio signals captured by the front-microphone of the listening smartphone.

Despite the constant magnitude sound source, received signals manifest substantial variations across the sampling indices. This verifies the intuition that waves with different frequencies experience different fading levels at the same location. Further, even around the same frequency, location 'A' and 'D' exhibit different fading profile — one may be deeply faded while the other experience peak signal strength. Hence, multipath fading effects are also location dependent.

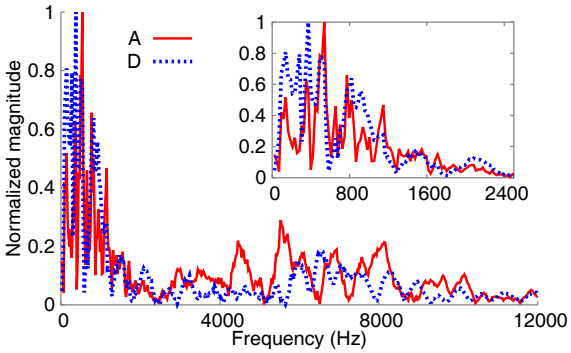


Figure 6: Amplitude spectrum density of two key strokes 'A' and 'D' on a wood table.

We further investigate the frequency-domain patterns of two actual keystrokes. We characterize such patterns by using the received signals' amplitude spectrum density (ASD). Suppose $R(t)(t = 0 \cdots T)$, are the discrete signal samples captured by the microphone, then the ASD is defined as: $FFT(R(t))$. Since audio signals are real numbers, their ASD is symmetric with respect to the half-frequency (*e.g.*, 24 kHz at 48 kHz sampling rate). We only use the first half to avoid duplication.

Figure 6 plots the ASD corresponding to user-generated click sounds at key locations 'A' and 'D', normalized with respect to the maximum across all frequency bins of each. We see that the majority of frequency components concentrate within 10 Hz and 1 kHz. The ASD of two locations peak at different frequency bins, and exhibit distinct values across frequencies. This provides us with a first hint for using ASD as location signature.

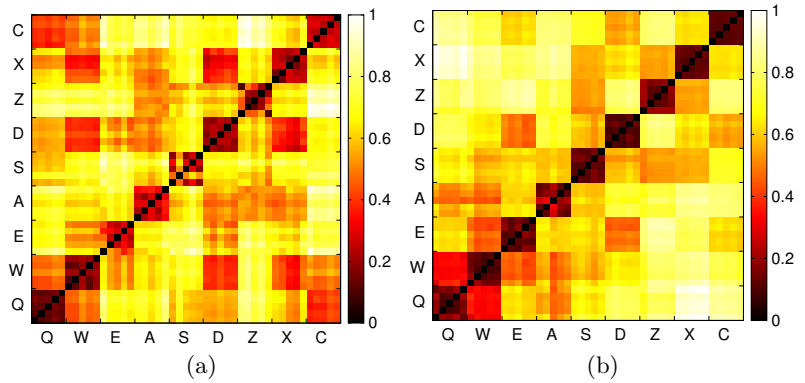
Summary of observation 1: *Multipath channel profile represents unique audio signatures for different keystrokes and enable signature-based localization.*

3.3 Spatial Granularity and Temporal Stability of Channel Profile

Consistency of ASD on the same key location Given the potential of ASD as location hint, one would ask: is ASD consistent across clicks of the same key, and will it be fine-grained enough to distinguish adjacent key locations? Figure 4(a) plots the Euclidean distance between the normalized ASD of 9 closely located keys. Each key is clicked by 5 times using finger tip and nail on a wood table, generating a total of 45 signatures.

We observe that short Euclidean distances are concentrated for each 5 keystrokes on the same location. Occasionally, a keystroke may have similar ASD with nearby or distant keys, yet the most similar ones almost always fall in the same location. Note that the diagonal line represent the zero distance between each key press and itself.

One may wonder if the ASD profile is possibly attributed to heterogeneous acoustic profile of the touch surface. To rule out this possibility, we place a speaker at the 9 key locations and repeat the chirp sounds 5 times each. Figure 4(b) plots the resulting Euclidean distances between the bursts of chirp tones. Now the ASD signatures show a more clear



(a) (b)

Figure 4: Euclidean distance between ASD of sounds at 9 key locations, each repeated 5 times in two cases: (a) each sound source is created by finger/nail clicking on the key locations; (b) the sound source is a chirp tone emitted from the key locations.

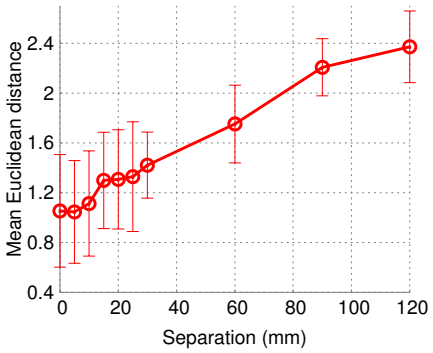


Figure 5: Euclidean distance between a group of keystrokes at a testing position and those at an anchoring position.

location-dependent trend. This confirms that there is no need to use heterogeneous surface materials. On the other hand, the better Euclidean distance profile of chirp tones implies that user's inconsistent click behavior can cause variation of the location signatures.

Spatial correlation of ASD signature We now examine in more detail how resilient ASD is to minor deviation of click positions, which can naturally happen because users' finger touch positions are not perfectly consistent. We first create an anchoring group of 25 clicks at a fixed position, and then generate testing keystrokes, which deviate from the anchoring position by 5 mm to 120 mm. For each testing position, we calculate the Euclidean distances of all (*testing, anchoring*) pairs, using ASD as the feature vector. Figure 5 plots the mean and standard deviation of the Euclidean distances. We observe that the Euclidean distance increases almost monotonically with the physical separation between keys. For physical separation of 5mm, the change of Euclidean distance is minor, implying that spatial correlation does exist between ASDs of closeby clicks. However, on average, keystrokes with physical separation of more than 1cm (roughly the distance between the center of one key and edge of a neighboring key) have larger ASD separation than those below 1cm. Therefore, clicks on the same key can be separated from those a neighboring key. Note that the Euclidean distances exhibit variance, mainly because the user cannot perfectly control the click positions.

Temporal stability of ASD To test temporal stability, we repetitively create groups of 10 clicks on the same key location. The experiment spans over one month, while the keyboard and phone are fixed on the same location. Ambient environment changes are minor, including placement/displacement of chairs, laptops, cups *etc.* Figure 7 plots the Euclidean distance between the ASD of a random keystroke in each group with that of all keystrokes in the first group. Over time, the mean Euclidean distance does not show significant change, although variance increases slightly after one month. Thus, compared with radio channels [17], the multipath profile of audio channels is more stable over time. It is also less sensitive to ambient objects movement, likely because reflected waves from those objects may be too weak and below the sensitivity of the microphone. We gauge the multi-path effects mainly come

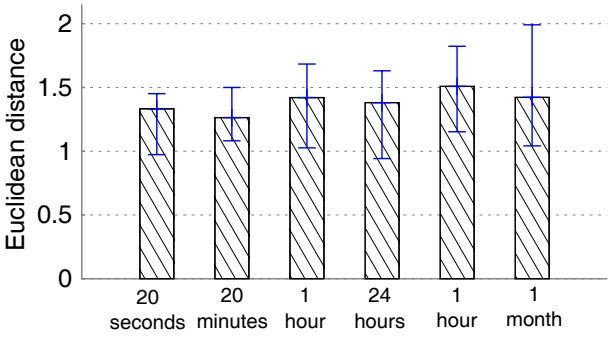


Figure 7: Euclidean distance between keystrokes varies negligibly over time. Error bars show the max-min values.

from keystroke sounds' propagation along the surface and reflections/diffractions around the smartphone body.

Summary of Observation 2: *The keystrokes' audio signatures follow consistent patterns within a key-size area, so the localization algorithm can be resilient to minor displacement of device and small variation of key-press positions. The signatures are also highly stable over time.*

3.4 Diversity from Multiple Microphones

Channel diversity from dual-microphone Inspired by the diversity gain in multi-antenna wireless communications, we examine whether dual-microphone, a standard equipment in modern smartphones, can enrich the ASD signature of a key location. Figure 8 plots the ASD of 10 consecutive clicks of the same location, received by two microphones on Galaxy Nexus. The ASD curves of different clicks show a highly consistent pattern on the same microphone. Yet across different microphones, they differ drastically. This implies that audio channel diversity does exist, even though the microphones separate at a much shorter distance (12.5 cm) than the half-wavelength (*e.g.*, 34.3 cm at 500 Hz frequency) of audible keystroke signals.

Coarse-grained localization in conventional dual-mic algorithms Microphone-array has been extensively used for sound source localization (SSL), *e.g.*, localization a speaker in a lecture room. The principle resembles human perception of sound direction, inferred by time-difference-of arrival

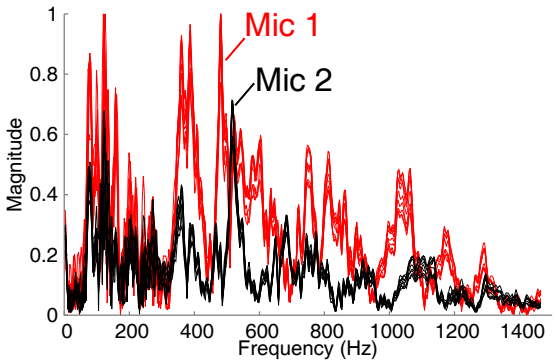


Figure 8: ASD of 10 key presses received by two microphones on the same smartphone. The ASD is normalized by the maximum across all frequency bins and keystrokes.

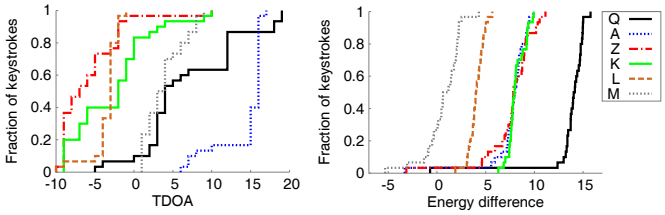


Figure 9: Distribution of the TDOA and EDIF between two microphones.

(TDOA) and energy difference (EDIF) between signals received by two ears [16].

Our earliest attempt to build UbiK followed this principle. Theoretically, on a 2D space, locations of the same TDOA value form two hyperbolas centered around the two microphones, and those of the same EDIF value form a circle centered around one microphone [18]. Intersections between these two form two points, but one can be eliminated as the microphones in UbiK always reside on one side of the keyboard. We compute the TDOA between two microphones using GCC-PHAT (generalized correlation with phase transform), a state-of-the-art algorithm widely used for SSL in practice [19]. The total energy received by each microphone can be derived by summing up the power spectrum density.

Figure 9 shows the distribution of TDOA and EDIF of each key, repetitively clicked by 20 times. Even for the same key, its TDOA values are inconsistent across clicks. TDOA algorithms assume a single non-reverberant path between sound source and microphone, hence it is highly sensitive to multipath reflections and minor deviation in click positions (which is unavoidable). Similarly, the EDIF of the same key spreads over a wide range, and even distant keys (*e.g.*, 'Z' and 'K') can have similar EDIF. Therefore, conventional dual-microphone SSL algorithms cannot achieve fine-grained localization as required by UbiK.

Summary of Observation 3: *Multiple microphones on mobile devices can provide spatial diversity and improve granularity of keystroke localization. The diversity mechanism should embrace, instead of avoid multipath reflections.*

4. KEYSTROKE DETECTION

UbiK's keystroke detection algorithm identifies the signals generated by key-presses. Its core design goal is to ensure (i) low false-alarm and mis-detection rate and (ii) resilience to noise coming from the user and ambient environment.

4.1 Basic Detection Mechanism

The audio signals produced by a key-press event manifests a common onset pattern, with a few outstanding peaks in the beginning followed by small reverberations, which together form a cluster of energy burst rising above noise. Since the printed keyboard is close to the mobile device, keystroke sounds are much stronger than ambient noises. UbiK leverages such unique profiles to single out the keystroke signals.

In its simplest form, the detection algorithm computes the received signal magnitude or power and declares a keystroke if it exceeds a threshold. Yet no different environment bears different noise levels. A keystroke detection mechanism must adaptively configure the threshold that separates keystrokes from noise. UbiK meets this goal by adapting the Constant False Alarm Rate (CFAR) algorithm [20], a statistical approach historically used in Radar systems to identify signals reflected by intruding objects.

CFAR approximates ambient noise power with a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. A significant energy burst is detected if the incoming signal power passes a threshold value of $(\mu + \gamma\sigma)$, or γ standard deviations above the mean noise floor. Given the noise distribution, the detection may also be a false alarm triggered by noise, with probability $1 - erf(\frac{\gamma}{\sqrt{2}})$, which decreases exponentially with γ.

In UbiK's CFAR implementation, we estimate the noise power μ through a moving average window with size W:

$$\mu(t) = \frac{1}{W}A(t) + (1 - \frac{1}{W})\mu(t-1),$$

where $A(t)$ is short-term average noise power right before t. Denote $x(t)$ as the received audio signal at time t, then,

$$A(t) = \frac{1}{W}\sum_{k=t-W+1}^{t} |x(t)|^2.$$

Estimation of variance $\sigma(t)$ follows the same way as $\mu(t)$. We choose the window size W to be much shorter than the duration of environment sounds, so that even if bursty interference occur, the interfering signals within W can still be approximated as Gaussian. An empirical value of 1 ms (48 samples at 48 kHz sampling rate) for W works across a wide range of environment according to our experiments.

UbiK declares a key-press event at t if: *(i)* incoming signal's energy $|x(t)|^2$ passes the CFAR threshold $(\mu + \gamma\sigma)$, and *(ii)* current detection separates from previous one by a safe margin. For devices with dual microphones, either one satisfying these conditions is accepted. We configure γ during the initial training stage, by linearly searching through all integer multiples of 0.5 within the range $[1, 10]$, and pick the highest one that results in zero mis-detections and false alarms for the known keystrokes. The safe margin between keys is configured to a value smaller than the minimum separation between two consecutive keytrokes.

4.2 Combating Bursty Noise

Since UbiK relies on energy based detection, it can be easily disturbed by bursty noise. However, the smartphone's microphones fall within a short-range to keystrokes, typically 10 to 20 centimeters, whereas sound signal power decreases with distance following a inverse-square law [16].

Therefore, far-field noise sources, *e.g.*, a few meters away from the microphone, cause negligible interference to keystroke detection unless they are exceptionally loud.

According to our field test of UbiK, near-field noise mainly comes from two sources: human voices nearby, and user tapping on the smartphone screen which causes non-trivial noise bursts due to close proximity to the microphones.

UbiK utilizes a sensor fusion technique to combat both noises. Gyroscope, which senses rotations in three dimensions, serves as a secondary source for keystroke detection in UbiK. At start-up time, UbiK assumes the phone remains stationary on the surface and collects gyroscope data for a short period for calibration. A threshold ϵ_1 for determining the existence of keystrokes, is then set at two times of the maximum of gyroscope readings during calibration. UbiK declares a keystroke only if both the audio and gyro confirm its presence. In such way, false alarms caused by human voice are eliminated since it does not cause surface vibrations.

Tapping actions on screen (caused by user's runtime feedback) disturb the gyroscope reading to a much larger degree compared to tapping on desk surface. A threshold ϵ_2 for tapping-on-screen is determined when user first taps the on-screen button to start training. ϵ_2 is empirically set to $\frac{1}{2}$ of the maximum gyroscope output during the user tapping. At run-time, only when the gyroscope readings are above ϵ_1 and below ϵ_2, a potential keystroke is confirmed.

Recent work has shown that accelerometers can detect keystrokes tapped on a physical keyboard [21]. In UbiK's usage case, smartphone's motion variations are much smaller. Since tapping bends the area closer to tapping point more than the area further away, a rotation of the smartphone can be captured more easily by gyroscope. Accelerometers, designed to measure larger movements, have more noises at such granular level. From our experiment using Galaxy Nexus and Nexus 7, gyroscope yields cleaner and larger distinctions between a steady phone and a shaking phone caused by key tapping.

5. KEYSTROKE LOCALIZATION

In this section, we describe how UbiK's keystroke localization algorithm leverages the previous experimental observations (Section 3) to distinguish the keystrokes.

5.1 Location Signature Design

Our measurements have established Amplitude Spectrum Density (ASD) as a promising location hints. ASD reflects the frequency domain acoustic channel profile caused by sound waves' multipath fading. Compared with the signal-power based fingerprinting widely used in geo-location systems, ASD incorporates a richer set of features, thus finer granularity. Compared with a time domain approach that uses the sound waves directly as fingerprint, ASD is insensitive to waveform ambiguities and unpredictable stretches. To synthesize such advantages and make ASD a practically useful signature, we still need to address the following problems.

Estimating keystroke duration UbiK uses D samples following the start of a keystroke to compute ASD. D represents an estimation of keystroke duration, which depends on surface type and users' click actions. We obtain D from the initial training setup. For each known keystroke in the training set, we first obtain the noise floor P_n preceding it (Section 4). A keystroke usually corresponds to a sudden jump in signal power that quickly reaches a peak level $P_{\max}$. Then, D is estimated as the number of samples between the start point and the point when mean signal power first drops to below $P_n + (P_{\max} - P_n) \times 10\%$. Capping the keystroke duration at a power level slightly (10%) above noise floor prevents UbiK from incorporating unnecessary noise following the actual keystroke signals. The estimated keystroke durations of all keystrokes in the training set are averaged to obtain D.

Optimizing frequency range Our previous measurement on a woodtable revealed that the main ASD features concentrate within a small frequency range. Adding higher frequencies into the signature increases the computational load during signature matching, which may worsen processing latency. Moreover, it may invite high-frequency noises that are not generated by the keystroke. The key problem here is how to determine the critical frequency range, which may vary on different solid surfaces. Our solution to this problem is inspired by the optimal separating hyperplane problem in statistical learning [22].

Conventionally, optimal separating hyperplane is used for binary classification, *i.e.*, finding a hyperplane that separates two classes of data and maximizes the distance to the closest point from either class. Denote $\boldsymbol{\beta}$ as the vector normal to the hyperplane, x_{i*} the vector of features in i-th data set, and $\boldsymbol{y}$ the correct prediction (either 1 or -1). Then the classification problem for a given x_{i*} can be cast as:

$$y_i = sgn(\boldsymbol{\beta x} + b) = sgn(\textstyle\sum_{j=1}^{N} \beta_j x_{ij} + b) \tag{1}$$

where $sgn(\cdot)$ is the sign function. The training process in binary classification solves the following optimization problem to obtain the optimal weights vector $\boldsymbol{\beta}$ and offset b [22]:

$$\arg_{\boldsymbol{\beta},b} \ \min ||\boldsymbol{\beta}||^2 \tag{2}$$

$$\text{s.t.} \quad y_i \left(\textstyle\sum_{j=1}^{N} \beta_j x_{ij} + b \right) \geq 1, i = 1, 2, \cdots, S_t \tag{3}$$

where S_t is the number of instances in the training set.

In UbiK, we cast the problem of finding the critical frequency range as a similar problem but with much lower complexity. Denote J as the "sweet spot" frequency below which the ASD features should be included. Since the frequency bins are not weighted, we only need to find:

$$\arg_{J,b} \ \min ||J||^2 \tag{4}$$

$$\text{s.t.} \quad y_i \left(\textstyle\sum_{j=1}^{J} x_{ij} + b \right) \geq 1, i = 1, 2, \cdots, S_t \tag{5}$$

This is a mixed-integer program, generally intractable. However, observing that there are only a limited number of possible values for J, we reformulate the optimization problem as a feasibility problem, and solve it by searching for the minimum J that results in a feasible value for b, such that:

$$b \geq \frac{1}{y_i} - \sum_{j=1}^{J} x_{ij}, i = 1, 2, \cdots, S_t \tag{6}$$

In the common cases, the critical frequency J is above 100 Hz. Thus we only need to start searching from the frequency bin: $B \cdot 100/F_s$, where F_s is the audio sampling frequency and B the FFT size (total number of frequency bins). In our actual implementation, by default $B = 4096, F_s =$

$48000, S_t = 3$ and the maximum search range is 5000 Hz. Thus, the maximum number of operations of Eq. (6) is $S_t \cdot B \cdot (5000 - 100)/F_s \approx 1254$, which can easily run in real-time on a modern mobile device.

Determining frequency resolution When computing ASD, a large B results in more frequency bins, thus higher frequency resolution, yet it also increases computational load. In UbiK, we empirically set B to the first 2's power larger than the keystroke duration. A typical keystroke (*e.g.*, on wood table or hard-cover paper) spans around 2000 samples (at 48 kHz sampling rate). Correspondingly, $B = 2048$.

5.2 Initial Training

During initial setup, UbiK displays instructions on the mobile device's screen to guide the user to sequentially click all keys (A–Z, 0–9, and symbolic/functional keys) on the printed keyboard. On a printed Apple Wireless Keyboard (AWK), with 56 keys in total (excluding PC-specific functional keys), this training process takes only around 1 minute, even for first-time users. Users are encouraged to use the same finger to click a key as they would in actual typing.

The audio samples generated by training keystrokes have known labels and are used for three purposes: *(i)* optimizing core parameters in UbiK's keystroke detection algorithm (Section 4); *(ii)* initializing and optimizing the ASD signatures as discussed above; *(iii)* providing benchmark ASD signatures to be used in keystroke localization.

Intuitively, repeating the training procedure multiple times can improve the keystroke localization accuracy. This entails more user workload. However, as shown in our evaluation (Section 8), it only takes 3 training samples each for UbiK to escalate its accuracy from 70% to above 91%. Such overhead is negligible if the keyboard is to be used for hours, *e.g.*, in a coffee shop, on an office desk or tray table of an airplane.

5.3 Localization Algorithm

UbiK's keystroke localization algorithm is a pattern classification scheme that matches the ASD features of user-typed keystrokes with those instances in the training set.

Basic classification After detecting a key-press event, UbiK extracts the ASD features from corresponding audio signals of both microphones, which together form a vector of length $2D$. Depending on the relative location of a keystroke, the two microphones may detect the event with different starting point. UbiK computes the ASD of the two sequences of signals, with the starting point of each separately, but keeping the same keystroke duration parameter D. This ensures the ASD features are best aligned in time and compared in a consistent manner.

Then, UbiK runs a nearest-neighbor based pattern matching (classification) algorithm that compares the extracted features with those in the training set. The training key with minimum distance is declared as the current keystroke and output to the user interface. In the simplest form, we use Euclidean distance as the metric of comparison.

Optimizing ASD features for classification Our controlled experiments (Section 3) have shown that click sounds on the same key location tend to have much shorter Euclidean distance (*w.r.t.* ASD features) than that between different keys. In practice, the Euclidean distance can be disturbed by multiple uncontrollable factors.

Recall that the ASD represents a mix of features from the click sound and the multipath channel distortion. The strength of user's clicks may vary over time, thus causing ASD variation even for the same key. However, the variation tends to simply scale the entire ASD curve. The frequency bin with maximum magnitude remains consistent. We thus normalize each ASD feature with its highest magnitude to improve resilience to variation of click strength.

Further, we observe that the variance of ASD feature within the same frequency bin (but across training instances) can reflect the confidence of pattern matching. For the same key, if the amplitude of a certain frequency bin exhibits a small variance, then that frequency bin should be considered as a highly reliable feature element. Thus, at run-time, for each frequency bin f of the user-typed keystroke, we scale its magnitude by $\frac{1}{V_f}$, where V_f is the magnitude *std.* of the training instances. Note that such scaling should be done before the above feature normalization. The frequency bin with peak magnitude is ignored since it tends to have standard deviation after normalization.

Fail-safe mode adaptation UbiK's keystroke detection algorithm can prevent false triggering by nearby bursty interferences, most commonly, human voice. However, a tougher case comes when human voice and keystroke sounds overlap, which contaminates the keystroke's ASD feature. UbiK tackles such cases by adapting to a fail-safe mode. Rather than outputting a wrong key value, which entails user correction and causes extra burden, UbiK outputs nothing but a "interference" warning on the user interface.

To identify such heavily interfered keystrokes, we observe that human speech tends to show a consistent amplitude for at least tens of milliseconds. In contrast, a keystroke features a cluster of high-amplitude signals, for a few milliseconds, followed by small vibrations. Therefore, whenever an energy bust is detected, UbiK takes the derivative of the signal amplitude envelop, starting from the highest peak and spanning one keystroke duration D. If the derivative's magnitude is below 50% than that of keystrokes in the training set, then UbiK decides the keystroke to be contaminated.

6. ONLINE CALIBRATION AND ADAPTATION

UbiK presumes the keyboard and mobile device are kept at stable positions throughout its usage life-cycle. In practice, the positions may be disturbed by, *e.g.*, surface vibration, screen touches, and user's repositioning of the mobile device. Over time, user's typing posture may also vary due to fatigue, rendering ASD features in the initial training set outdated. UbiK employs a run-time calibration and adaptation framework to tackle such problems.

6.1 Runtime Calibration

UbiK executes run-time calibration by combining user correction with its own localization hints. For each keystroke, besides the output from the localization algorithm, it also displays the top 5 *candidate keys*, *i.e.*, those with shortest feature distance. User can click a candidate if it is the actual intended key. In the rare case when the candidate list does not contain the intended key, the user can reenter the key using the built-in on-screen keyboard. UbiK places the

"Delete" key on the screen instead of the printed keyboard, since it must be reliably recognized for calibration purpose. To minimize disturbance to the ASD features, the mobile device should remain on the surface when user performs on-screen correction.

6.2 Adapting and Optimizing Training Set

UbiK updates the training data set progressively while the user types in more keys. The update opportunistically employs feedback hints about correctness of a localization decision. UbiK deems a localization output as correct if the user does not execute run-time correction. Since the user may not immediately correct a character error, UbiK defers the decision on correctness until then end of the current word input (using space and punctuation as a hint). Besides, user tapping a character on the candidate list implies that an localization error occurred. Notably, user pressing the "Delete" button is not necessarily a hint for localization error because it may be the user's own input error.

If a keystroke is correctly located, the corresponding ASD feature will be put into that key's training set as a new training instance. In addition, UbiK employs a feedback based weighting mechanism to rank the significance of existing training instances.

Weight design for correctly located keys. At time t, UbiK associates a weight $w_{ki}(t)$ to the i-th instance of key k in the training set. t is discrete and simply counts the number of localization runs. $w_{ki}(0)$ equals 1 for all instances. Suppose $d(k, i, S(t))$ denotes the distance metric between each training instance and the incoming key features $S(t)$, then UbiK uses $d(k, i, S(t)) \cdot w_{ki}(t)$ as the distance metric to decide the nearest training instance for $S(t)$.

If a keystroke is correctly located, the corresponding training instances decreases its weight as:

$$w_{ki}(t+1) = V(w_{ki}(t)) \tag{7}$$

where $V(\cdot)$ is a convex function, such that the decreasing step becomes smaller if $w_{ki}(t)$ is already small. To prevent a small set of instances biasing the classification, $w_{ki}(t)$ is capped between 0.8 and 1.2. Accordingly, we set

$$V(x) = 0.8 + (x - 0.8)^2 \tag{8}$$

to ensure convexity within this range.

Weight design upon localization error. If a keystroke is located wrongly, the nearest training instance decreases its weight as:

$$w_{ki}(t+1) = X(w_{ki}(t)) \tag{9}$$

where $X(x)$ is a concave function, designed following similar intuition as $V(x)$, as:

$$X(x) = 1.2 - (x - 1.2)^2 \tag{10}$$

In addition, after user enters the correct key, the nearest instance to that key will update its weight following Eq. (8). To prevent bias by frequently used keys, we further normalize $w_{ki}(t)$ by t, *i.e.*, the number of times instance i is updated. Here the frequency of usage means the frequency when a training instance is updated, which in turn depends on how frequently the corresponding key is pressed. Without the normalization operation, a frequently used key, say 'A', may have training instances with very small weights, which results in small Euclidean distances between 'A' and all other keys, thereby causing localization errors.

7. IMPLEMENTING UbiK ON ANDROID

We implemented UbiK as a standalone application program running directly on Android devices. Specifically, we implement all the components described in Section 4, 5 and 6. In implementing the online adaptation, we eliminate training instances with largest weights, and keep a constant training set size of 10. We found the user may occasionally miss the localization error and the corresponding instance will be mistakenly put into the training set. However, such instance does not pollute the training set in a significantly way, because it does not contribute to correct localization and thus will be replaced from the training set in a short time with online adaptations.

As for dual-microphone audio acquisition, the Android application framework blocks the stereo recording (the back microphone is only used for noise cancellation by the framework). We overcame this constraint by bypassing the build-in framework and using the low-level tinyalsa driver instead. We modified the tinyalsa driver so that it can stream dual-channel audio samples to applications through standard I/O. UbiK's Java implementation triggers the recording by forking tinyalsa as a child process. The implementation gets recorded samples from tinyalsa every 10 ms. These samples are then put into a 100 ms audio buffer. Our keystroke detection algorithm runs on these 10 ms audio instances. When a keystroke occurs, the application continues to collect audio until enough samples are obtained for extracting ASD signatures.

To benchmark the run-time efficiency of UbiK in our implementation, we use a Galaxy Nexus smartphone to record a keystroke sound, and replay it while running UbiK. Then we measure the latency between the time when the faked keystroke sound is played back and when UbiK outputs the localization result on the screen. We found an average processing latency of 51.4 ms, and standard deviation 2.7 ms. Such latency is well below human response time. In fact, we experience no lagging effects when using UbiK.

8. SYSTEM EVALUATION

In this section, we first evaluate each design component of UbiK, as well as the underlying impact factors in a variety of test scenarios. We then conduct a user study to verify the effectiveness and usability of UbiK in comparison with existing text-entry methods.

8.1 Micro-benchmark Tests

We run experiments using the following default setting unless explicitly specified. We use a Galaxy Nexus phone, which is placed near the edge of a printed keyboard on top of a wood table. We use AWK as the default keyboard and test design components with online-adaptation disabled (except in the adaptation test).

Accuracy of keystroke detection We first evaluate UbiK's keystroke detection in four scenarios, specifically, in an office environment (wood table), a server room (metal cabinet), at food court (wood table) and on a flying airplane (tray table). The former three represent typical, daily environments such as office and cafeteria, and the latter is used to examine UbiK in an extremely noisy environment. These our test scenarios reflect a variety of realistic noise levels, which ranges from 23.2 to 76.5 dB. Noise level is measured by Sound Meter Pro, an Android app calibrated by a

	Office	Server room	Food court	Airplane
Noise level	23.2 dB	45.8 dB	41.0 dB	76.5 dB
P_{mis}	0.33%	1.33%	0.33%	1.67%
P_{fls}	0.0%	0.0%	0.67%	5.0%
P_{mis} (no gyro)	0.0%	2.0%	4.0%	8.67%
P_{fls} (no gyro)	0.67%	0.33%	7.33%	8.67%

Table 1: Keystroke detection accuracy in four environments.

	Office	Server room	Food court	Airplane
Loc. accuracy	97.1%	94.0%	91.9%	92.4%

Table 2: A baseline accuracy test of keystroke localization in four environments.

professional acoustic meter [23]. In each test, a user uses finger tip plus nail margin to make 300 clicks, each on a randomly selected key position. The click strength is empirically maintained to be audible by the user, at a similar level as PC keyboard click sound. Across all the experiments, the detection algorithm uses the default set of parameters described in Section 4.

Table 1 presents the resulting mis-detection (P_{mis}) and false-alarm (P_{fls}) rates. It reveals that the keystroke detection is accurate, robust and reliable. Here are two observations. First, the error rates are kept below a reasonable level ($< 5\%$ in the worst case). In the relatively quiet environment (office and food court), both P_{mis} and P_{fls} are negligible. On an extremely noisy airplane with occasional vibration, they only increase to 1.67% and 5%, respectively. Second, gyroscope improves the accuracy, especially in case of noise. The noise level is lower at food court than in the server room, but P_{fls} can be up to 7.33% without gyroscope. This is mainly attributed to the interference from nearby human voices. Falsely detected keys trigger a chain effect, leading to subsequent miss detections (recall the safe margin between keystrokes). As a result, error rate becomes unacceptably high when gyro is disabled. Note that, in a quiet environment, CFAR based detection maintains an error rate below 2%, even with gyro sensor disabled. Clearly, UbiK is able to effectively eliminate the negative impacts of noise and yield a robust and accurate detection.

Accuracy of baseline localization We perform a baseline test of keystroke localization in the above environments. The keyboard setting remains the same as above, except that all 56 keys on the AWK are pressed sequentially, each repeated 25 times. We carry on a conventional leave-one-out cross-validation. This statistic tends to be generous, since training and testing datasets are collected from the same user and adjacent in time (which will naturally tend to be more similar). Nonetheless, it provides a micro-benchmark to validate the effectiveness of UbiK's location signature design and optimization. Note that add-on features such as online-adaptation and calibration are disabled.

Figure 10 plots the resulting confusion matrix in the office room. Each element (i, j) represents the probability that key i's nearest neighbor is one of the clicks on key j. Clearly, keystrokes localized by UbiK densely overlap with the actual ones. Among those few erroneous localization results, most are mistaken with one or two other keys. Intuitively, such errors can be further reduced or eliminated by online adaptation (the training set is updated as user inputs more keys).

Table 2 enumerates the average localization accuracy in different environment, where false detection and miss detection are manually eliminated in order to isolate the impact of keystroke detection. The results demonstrate remarkably high accuracy, above 97% in office environment, and around 92% in adverse acoustic environment.

Impact of initial training The initial training set size affects UbiK's usability and accuracy — a tradeoff that deserves a fine-balance. Figure 11 plots the achieved accuracy as the number of initial training instances increases. It shows that localization accuracy is around 70% even with one initial training. As the number of training instances grows up to 3, accuracy escalates to above 91% on average. Further increasing the number beyond 5 provides marginal improvement only. Therefore, the user only needs to input 3 training instances per key to achieve reasonable performance. This is a small burden to pay if the keyboard life cycle spans more than a few minutes but may be undesirable otherwise. Later we will show that the training can be embedded in subsequent typing to reduce user work load at the start.

Effectiveness of frequency range optimization Our initial implementation of UbiK used an empirical frequency range of 0 to 5 kHz — roughly the same as that of human voice — in the ASD feature selection. This worked well if the user carefully stays consistent in terms of finger gesture, click position and strength, when making the keystrokes. However, the performance becomes erratic once she types rapidly. Figure 12 plots the localization accuracy resulting from this empirical approach, in comparison with that after UbiK's frequency-range optimization mechanism (Section 5.1). The experiments run on four different types of solid surfaces, each repeated 8 times (std. shown by error bars). UbiK can maintain above 95% of accuracy across all the experiments, whereas the empirical frequency setting achieves only around 80%.

Resilience to keyboard/phone displacement Recall that the ASD features are coherent within about a key-sized area. Thus minor displacement of the mobile device or printed keyboard should be tolerable.We investigate this intuition by placing the smartphone in various ways and test the effectiveness of the online-adaptation algorithm. The resulting impact on accuracy is shown in Figure 13. Each sample counts the localization accuracy averaged over past 50 keystrokes. Localization accuracy may drop to around 80% if the phone is moved by one key's edge size. With less displacement (1/3 or 1/2 key), the decrease is much smaller. Occasionally, accuracy can be disturbed by other factors, such as user clicking the boarder between keys. However, in all these cases, UbiK's online adaptation scheme can quickly restore the accuracy to above 95% after a few tens of inputs. When user moves the phone away and tries to restore it to the original position ("best realignment"), the accuracy is virtually unaffected. We expect the best realignment to be a common case user may encounter in practical usage of UbiK.

Fail-safe mode under bursty interference Bursty interferences pose a greater challenge than noises with a high but stable power. UbiK's fail-safe adaptation strives to alleviate the impact of bursty noise. Figure 14 verifies the effectiveness of this approach, where we use a speaker to play

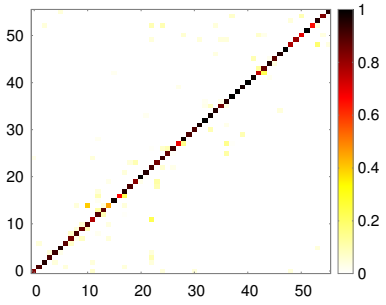


Figure 10: Confusion matrix of 56 keys on AWK.

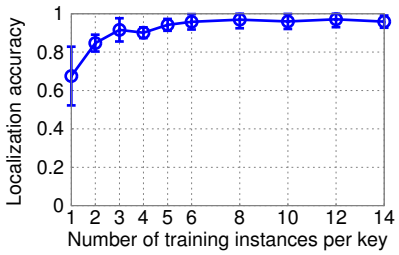


Figure 11: Impact of initial training set size. Error bars show std. across 8 experimental runs.

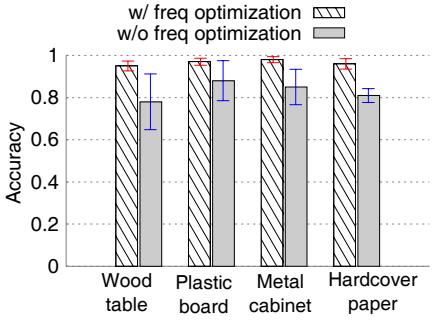


Figure 12: Impact of frequency range optimization when running UbiK on different surfaces.

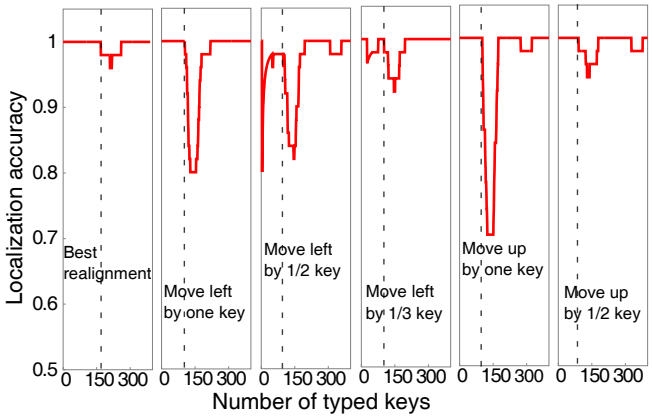


Figure 13: UbiK's online calibration and adaptation helps to quickly restore its high accuracy after a phone displacement. Dotted lines denote the time when displacement occurs.

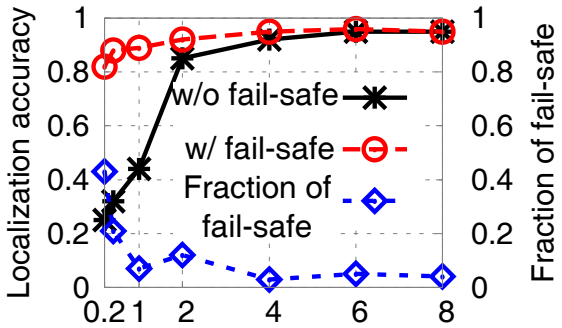


Figure 14: UbiK alleviates the impact of bursty interference by adapting to the fail-safe mode.

	US ANSI	UK ISO	AWK	Split
Accuracy	96.0%	92.5%	94.7%	98.2%

Table 3: Localization accuracy on different keyboard layouts.

human voice and vary its distance to the keyboard to create different levels of interference. Fail-safe mode isolates the keystrokes polluted by bursty interference, thus maintaining above 81% of localization accuracy even if the interferer is 20 cm away from the keyboard. Accuracy increases to above 89% as the interferer moves beyond 1 meter. In contrast, accuracy is degraded to as low as 22% without fail-safe adaptation. Notably, the fraction of keystrokes that are muted by the fail-safe adaptation can be 4% to 11% when the interference source is 1 or 2 meters nearby, which may result in undesirable experience for the typist.

Impact of keyboard layout We test UbiK on four keyboard layouts: US ANSI, UK ISO, AWK and Split keyboard. All keyboards are scaled on letter-size paper while maintaining the length/width aspect ratio. After scaling, the inter-key distance is comparable (1 to 2 mm shorter) to a physical keyboard like AWK. Table 3 shows the average localization accuracy. Keyboard layouts cause at most 5.7% of performance variation. In general, keyboards with slightly smaller key sizes (*e.g.*, US ANSI and AWK) performs better than those with larger ones (*e.g.*, UK ISO). This is mainly because of less variation in click positions. Similarly, keyboards with larger key separation (*e.g.*, the split keyboard) outperform others. Nonetheless, all keyboards experience an accuracy of above 92%, and have space to be improved after augmenting online adaptation.

Power consumption We have used the Monsoon mobile power monitor [24] to profile the power cost of UbiK. Specifically, we put a Galaxy Nexus phone in three states: (1) idle with display on; (2) running UbiK but without key input; (3) running UbiK with fast typing (more than 2 characters per second). Each state is maintained for 1 minute. The resulting average power consumption in each state is 1049.3 mW, 1160.8 mW, 1244.0 mW, respectively. Thus, UbiK incurs an additional 194.7 mW on top of the base power consumption (18.5% of power cost). To put the statistics in perspective, we also conducted measurement when browsing a CNN website through WiFi, which results in 1233.8 mW of power consumption — comparable to that of UbiK in active typing mode.

Impact of mobile device models Besides Galaxy Nexus, we have tested UbiK on alternative hardware platforms: Galaxy Note and Nexus 7. Both block dual microphone recording from firmware level. So we can only test with a single microphone. We find the keystroke localization accuracy on different devices varies slightly, possibly due to varying microphone quality. On Nexus 7, even with a single microphone enabled, the accuracy is comparable to Galaxy Nexus with two microphones. The Galaxy Note's accuracy is only 1–3% lower, under the default test setting. Notably, for Galaxy Nexus, its accuracy can drop by around 5% if a

	U1	U2	U3	U4	U5	U6	U7
PC	1000	2000	3000	1000	500	1000	1000
Onscreen	800	100	1500	100	100	500	500
UbiK	$> 10^4$	> 5000	$> 10^3$	New	New	New	New

Table 4: User reported proficiency with PC, Onscreen (average number of characters per day) and UbiK (total characters tried). All users have little experience with Swype except U3, who inputs > 2000 characters per day using Swype.

single microphone is used. We plan to test UbiK on other device models in future.

8.2 User Study

To evaluate the usability of UbiK in practice, we develop a standard text-entry field trial to compare our approach to others in a user study.

Experiment setup Seven participants (2 females and 5 males) are recruited from our university. They ran UbiK in several different environment, including home, library and office. Each participant completes four sessions, each involving typing regular text sentences and random characters. The random characters are uniformly selected from A-Z, digits and symbols on the AWK. The text sentences are randomly picked from the standard MacKenzie set [25], which well represents the usage frequency of English characters and words. Each sentence begins with a numerical index and ends with a random punctuation (, or .).

In the user study, we compare UbiK with three other popular input methods: a Dell PC keyboard, Google's Android on-screen keyboard, and Swype [26], which allows the user to enter words by sliding a finger across characters, and then uses a language model to guess the intended word. Before using each input method, the user is given a 10-minute warm-up period to familiarize themselves with the keyboard. Users are given the freedom to choose the solid surface from wood table, hard-covered paper, plastic board, and metal cabinet, as they prefer. The whole study is run in an office environment.

In all trials, participants are instructed to type as they do on a physical keyboard. They are allowed to correct erroneous input as they go. However, if they are unaware of a mistake until several characters later, they then should ignore the mistake and continue. This imitates occasional typos on a physical keyboard [25]. We evaluate the fraction of residual errors as well as the number of characters (including space and enter keys) per second.

Text input Table 4 lists user proficiency with various keyboard input methods. It is based on the interviews before the study. Figure 15 plots user performance when they enter the benchmark text. Two performance metrics of input speed and error rate are measured.

We make three observations. First, UbiK improves the input speed in real use. For users with less on-screen keyboard experience (*e.g.*, U2, U4 and U5) , their input speed can be more than doubled with UbiK. For U1 who is the most proficient UbiK user, even though she is also a heavy on-screen keyboard user, her input speed is improved by 83% with UbiK, with only slight increase of error rate (around 2%). Notably, two users (U6 and U7) had short nail margins, and struggled to maintain high input accuracy, and thus they do not witness much improvement with UbiK.

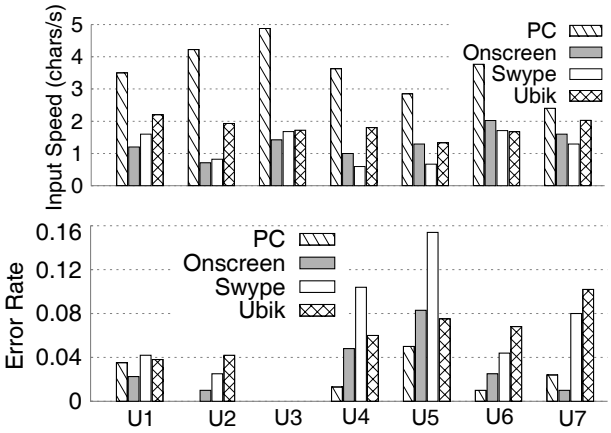


Figure 15: Text entry performance of different users with different keyboards.

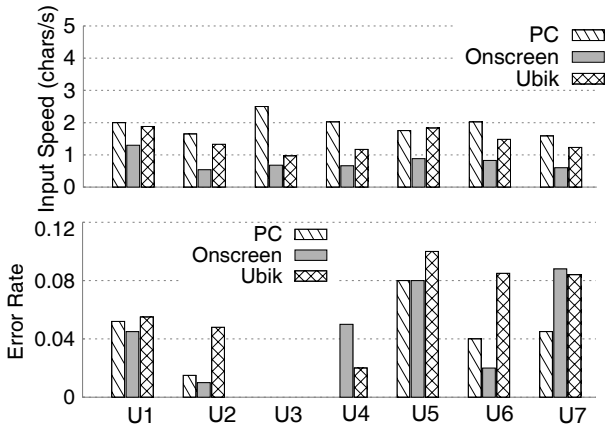


Figure 16: Random text entry performance of different users with different keyboards. Swype performs the same as Onscreen as dictionary is not applicable.

Second, UbiK is easy to use. After a quick warm up, four new users (U4-U7) can type 1.2–2 characters per second, slightly smaller than the proficient users.

Third, accuracy is relevant (sensitive) to user proficiency. The four new users tend to make more mistakes. We gauge they are less familiar with input tricks. Another factor is their personal behaviors; they seldom identify and correct typing errors immediately. We admit UbiK is relatively more erroneous than other methods. Yet as a new technique, its error rate is still comparable and tolerable. Moreover, it is promising to further lower its error rate (as U3 does). The Swype input method does not noticeably improve performance compared with on-screen, primarily because most users are unfamiliar with it and needs to waste time pondering about how to move their fingers.

Random character input Figure 16 shows the results of random text input. As the text involve a substantial amount of digits and punctuations, the on-screen keyboard suffers from high latency in switching between character/symbol keyboards. Thus, UbiK easily outperforms on-screen keyboard for all users. Note however in both text and random input, UbiK still lags far behind the conventional PC keyboard, partly because of user proficiency, and partly because of the overhead when users attempt to calibrate localization errors from UbiK.

9. DISCUSSION

As a first attempt to realize mobile text entry through fine-grained localization, UbiK still bears several limitations that worth further investigation.

Usability Although the size of the printed keyboard is the same as a physical keyboard, it cannot emulate the kinesthetic and tactile feedback that PC users feel. Existing studies targeting such PC typists pointed out that the removal of feedback significantly reduces the typing performance [27]. Thus, lack of such feedback is another reason why UbiK's input speed is lower than that of a PC keyboard. We also find that inexperienced users pay substantial visual attention on the printed keyboard to navigate their fingers to the correct key position, which further reduces input efficiency.

Tradeoff between accuracy and input speed Localization accuracy has been our primary objective in UbiK. We observe that even for first-time users, accuracy can be above 95% if the user is encouraged to keep consistent keystroke patterns, *e.g.*, always using nail margin plus finger tip to click the keys. However, for inexperienced users, this imposes mental and behavioral burden and hampers input speed. This is the primary reason why the input speed is incomparable to a PC keyboard. An immediate solution is to augment UbiK with a dictionary based error-correction model. This is likely to boost robustness to typing inconsistencies, thus improving input speed.

Robustness in keyboard choices and key clicking It should be noted that UbiK does not rely on the consistency of click strength to distinguish keys. As mentioned in Section 5.3, we normalize each ASD feature vector with its highest-magnitude element so that the keystroke detection can be resilient to the variation of click strength. After the normalization, the frequency-domain ASD features are highly distinguishable (Section 3).

Despite users are encouraged to use the same finger to click a key as they would in actual typing (Section 5.2), it is not mandatory. Such operation improves consistency between the training keystrokes and actually typed keystrokes, thereby improving the typing accuracy. However, UbiK still achieves high accuracy in our experiments even if different fingers are used to press the same key, as long as there is some consistency (*e.g.*, keep using nail tip to press the key).

Security concerns High accuracy in keystroke localization might raise security concerns. An attacker may be able to decipher a user's keystrokes by eavesdropping the keystrokes and stealthily training UbiK on the keyboard. To mitigate it, one possible shield is to use an randomized order for initial training. Without knowing the exact mapping between the ASD features and the keys, it turns much harder (*e.g.*, it requires a large amount of eavesdropping samples) to infer the corresponding keys. Developing counter-measures to such attacks are beyond the scope of our current work.

Other mobile devices Due to lack of hardware, we mainly used smartphones throughout our tests. UbiK is likely to make more difference for small wearable devices like smart watches and glasses, which we will explore in future.

10. RELATED WORK

Fine-grained wireless localization. Fingerprinting based localization, the basic idea behind UbiK, is known to achieve fine-granularity compared with timing-based approaches in wireless location frameworks. Recent measurement study [9] revealed that a combination of WiFi access points and FM broadcast stations' RSS signatures can enable localization accuracy at 1 foot level. Wireless multipath fading profile can be extracted using sophisticated virtual antenna arrays, and used as location signature for objects attached with RFID tags [14]. Frequency-dependent fading characteristics have also been employed [17] for indoor localization with 1 meter accuracy. In contrast to these wireless location solutions, UbiK represents the first work to achieve ultra fine-grained, centimeter scale localization. Further, UbiK cannot take advantage of any well-designed beacon signal patterns. Instead, it faces the unreliability and variation of click patterns even for keystrokes on the same location.

Acoustic ranging and sound source localization. UbiK is reminiscent of the classical sound source localization problem, which has a wide range military, scientific and commercial applications. Due to long propagation time, audio signals' TDOA or directional-of-arrival can be easily measured using a microphone array [10]. However, as verified in Section 3, such non-parametric solutions are extremely vulnerable to indoor reverberation effects and unsuitable for fine-grained localization. Using active audio beacons, it is feasible to achieve localization accuracy on the order of several centimeters [11, 28]. Unfortunately, UbiK's keystrokes cannot be generated using audio beacons.

Keyboard eavesdropping. UbiK is closely related with recent works in keyboard emanation. Asonov *et al.* [29] investigated acoustic emanations of a PC keyboard generated by click sounds. FFT results of keystroke signals are directly used as features to train a neural network. However, even with 100 trainings per key, the approach can only achieve 79% accuracy. The problem is revisited in [30] using an unsupervised learning approach, which heavily rely on dictionary and is unsuitable for real-time keystroke recognition. SpiPhone [21] uses sensing data from accelerometers to decipher keystrokes, base on an artificial neural network. Similar to [29], substantial training (150 instances per key) is needed and best accuracy is only 80%. However, since these approaches mainly target security/privacy, even a low level of accuracy may raise alarming problems.

Customized keyboard for mobile devices. Text-entry method has been an active research area in mobile human-computer interaction. Substantial efforts have been devoted in redesigning the keyboard to improve usability, *i.e.*, by adaptively zooming the keys [2, 5], rearranging characters, leveraging context information or additional virtual space on the mobile device [3, 4]. A class of projection based mobile keyboards have been studied in the past decade of HCI research [6–8, 31]. They use an infrared or visible light projector to cast a keyboard on a surface, and then run sophisticated optical ranging or image recognition algorithms to identify the keystroke. Since additional hardware platforms are needed, such solutions are not yet ready to solve the keyboard bottleneck for mobile devices.

Acoustic touch sensing has been exploited recently in novel human-computer interaction applications. TapSense [32] extracts acoustic features from different part of fingers to create additional dimensions of input information. Touch&Activate [33] enables touch sensing on everyday objects, again through acoustic signals collected from closely attached microphones. The interactive window project and follow-on works [34, 35] localize clicks on hard surfaces using surface-

mounted high sampling-rate microphones. It is unknown if such approaches work with COTS devices.

11. CONCLUSION

In this paper, we have designed and implemented UbiK, which enables a novel text-input solution for mobile devices via keystrokes on external, solid surfaces. UbiK is grounded on experimental evidences that verify the feasibility of fine-grained, centimeter-scale sound source localization, by using the multipath channel profile as location signatures. These observations are consolidated in a complete system design that realizes accurate detection and localization of keystrokes, and online adaptation of keystroke signatures based on user feedback. Our evaluation of UbiK demonstrates around 95% of localization accuracy across a variety of settings. A field trial involving new and experienced users shows that UbiK can significantly outperform current on-screen keyboards in terms of input efficiency, with slight increase of error rate. Although a physical keyboard is clearly preferable, UbiK provides a viable means for small mobile devices to support text entry.

12. REFERENCES

[1] M. Kölsch and M. Turk, "Keyboards without Keyboards: A Survey of Virtual Keyboards," in *Proceedings of Sensing and Input for Media-centric Systems*, 2002.

[2] K. Al Faraj, M. Mojahid, and N. Vigouroux, "BigKey: A Virtual Keyboard for Mobile Devices," in *Proceedings of International Conference on Human-Computer Interaction*, 2009.

[3] M. Goel, A. Jansen, T. Mandel, S. N. Patel, and J. O. Wobbrock, "ContextType: Using Hand Posture Information to Improve Mobile Touch Screen Text Entry," in *Proc. of ACM CHI*, 2013.

[4] O. Schoenleben and A. Oulasvirta, "Sandwich keyboard: Fast ten-finger typing on a mobile device with adaptive touch sensing on the back side," in *Proc. of ACM MobileHCI*, 2013.

[5] S. Oney, C. Harrison, A. Ogan, and J. Wiese, "ZoomBoard: A Diminutive Qwerty Soft Keyboard Using Iterative Zooming for Ultra-small Devices," in *Proc. of ACM CHI*, 2013.

[6] H. Du, T. Oggier, F. Lustenberger, and E. Charbon, "A Virtual Keyboard Based on True-3D Optical Ranging," in *Proceedings of the British Machine Vision Conference*, 2005.

[7] H. Roeber, J. Bacus, and C. Tomasi, "Typing in Thin Air: The Canesta Projection Keyboard - a New Method of Interaction with Electronic Devices," in *ACM CHI Extended Abstracts*, 2003.

[8] C. Harrison, H. Benko, and A. D. Wilson, "OmniTouch: Wearable Multitouch Interaction Everywhere," in *Proc. of ACM UIST*, 2011.

[9] Y. Chen, D. Lymberopoulos, J. Liu, and B. Priyantha, "FM-based Indoor Localization," in *Proc. of ACM MobiSys*, 2012.

[10] T. Gustafsson, B. Rao, and M. Trivedi, "Source Localization in Reverberant Environments: Modeling and Statistical Analysis," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, 2003.

[11] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan, "BeepBeep: a High Accuracy Acoustic Ranging System Using COTS Mobile Devices," in *Proc. of ACM SenSys*, 2007.

[12] K. Ho and M. Sun, "Passive Source Localization Using Time Differences of Arrival and Gain Ratios of Arrival," *IEEE Transactions on Signal Processing*, vol. 56, no. 2, 2008.

[13] S. Sen, J. Lee, K.-H. Kim, and P. Congdon, "Avoiding Multipath to Revive Inbuilding WiFi Localization," in *Proc. of ACM MobiSys*, 2013.

[14] J. Wang and D. Katabi, "Dude, Where's My Card?: RFID Positioning That Works with Multipath and Non-Line of Sight," in *Proc. of ACM SIGCOMM*, 2013.

[15] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[16] M. Vorlander, *Fundamentals of Acoustics, Modelling, Simulation, Algorithms and Acoustic Virtual Reality*. Springer, 2008.

[17] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, "You Are Facing the Mona Lisa: Spot Localization Using PHY Layer Information," in *Proc. of ACM MobiSys*, 2012.

[18] W. Cui, Z. Cao, and J. Wei, "Dual-Microphone Source Location Method in 2-D Space," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2006.

[19] J. H. DiBiase, "A High-Accuracy, Low-Latency Technique for Talker Localization in Reverberant Environment," Ph.D. dissertation, Brown University, 2000.

[20] R. Blum, S. Kassam, and H. Poor, "Distributed Detection With Multiple Sensors I. Advanced topics," *Proceedings of the IEEE*, vol. 85, no. 1, 1997.

[21] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(Sp)iPhone: Decoding Vibrations from Nearby Keyboards Using Mobile Phone Accelerometers," in *Proc. of ACM CCS*, 2011.

[22] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2013.

[23] Smart Tool Co., "Sound Meter Pro." [Online]. Available: https://play.google.com/store/apps/details?id=kr.aboy.sound

[24] Monsoon Solutions, Inc., "Monsoon Power Monitor," http://www.msoon.com/LabEquipment/PowerMonitor/.

[25] I. S. MacKenzie and R. W. Soukoreff, "Phrase Sets for Evaluating Text Entry Techniques," in *ACM CHI Extended Abstracts*, 2003.

[26] Swype Inc., "The Swype Virtual Keyboard," 2013. [Online]. Available: http://www.swype.com/

[27] S. Kim, J. Son, G. Lee, H. Kim, and W. Lee, "TapBoard: Making a Touch Screen Keyboard More Touchable," in *Proc. of ACM CHI*, 2013.

[28] Z. Zhang, D. Chu, X. Chen, and T. Moscibroda, "SwordFight: Enabling a New Class of Phone-to-phone Action Games on Commodity Phones," in *Proc. of ACM MobiSys*, 2012.

[29] D. Asonov and R. Agrawal, "Keyboard Acoustic Emanations," in *IEEE Symposium on Security and Privacy*, 2004.

[30] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard Acoustic Emanations Revisited," *ACM Transactions on Information System Security*, vol. 13, no. 1, 2009.

[31] J. Mantyjarvi, J. Koivumaki, and P. Vuori, "Keystroke Recognition for Virtual Keyboard," in *Proc. of IEEE International Conference on Multimedia and Expo (ICME)*, 2002.

[32] C. Harrison, J. Schwarz, and S. E. Hudson, "TapSense: Enhancing Finger Interaction on Touch Surfaces," in *Proc. of ACM UIST*, 2011.

[33] M. Ono, B. Shizuki, and J. Tanaka, "Touch & Activate: Adding Interactivity to Existing Objects Using Active Acoustic Sensing," in *Proc. of ACM UIST*, 2013.

[34] J. A. Paradiso, C. K. Leo, N. Checka, and K. Hsiao, "Passive Acoustic Knock Tracking for Interactive Windows," in *ACM CHI Extended Abstracts*, 2002.

[35] D. T. Pham, Z. Ji, M. Yang, Z. Wang, and M. Al-Kutubi, "A Novel Human-computer Interface Based on Passive Acoustic Localisation," in *Proc. of International Conference on Human-computer Interaction*, 2007.