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

J.J. Wang, Kaichen Zhao and Xinyu Zhang

University of Wisconsin-Madison

Chunyi Peng

The Ohio State University

Touch screen is a bottleneck of mobile devices

- Touch screen is shrinking. Fingers won’t change

- Typing can be painful.

Existing Solutions -- Hardware

Bluetooth keyboard

Apple Wireless Keyboard $69

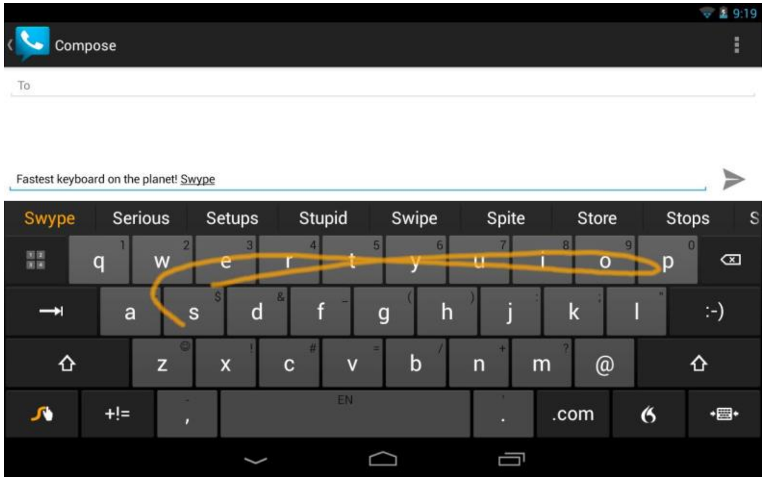
Projection keyboard

Celluon Laser Projection Keyboard $79.99

Existing Solutions -- Software

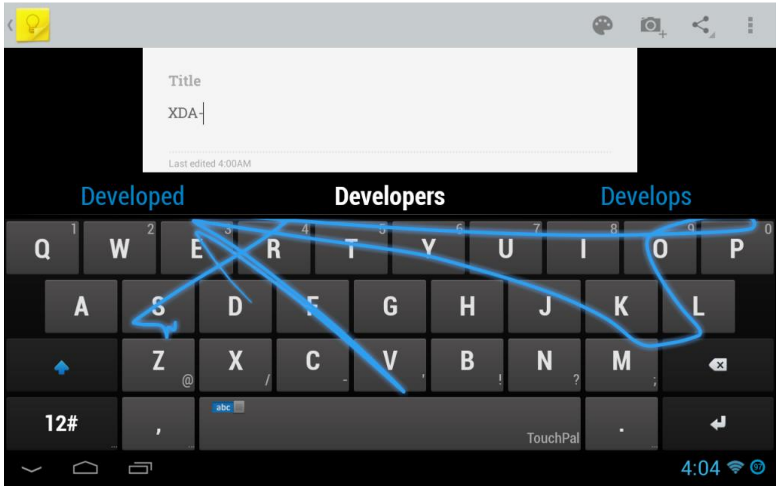
SwiftKey/Swype

Word



TouchPal X

Sentence



UbiK: The Ubiquitous Keyboard

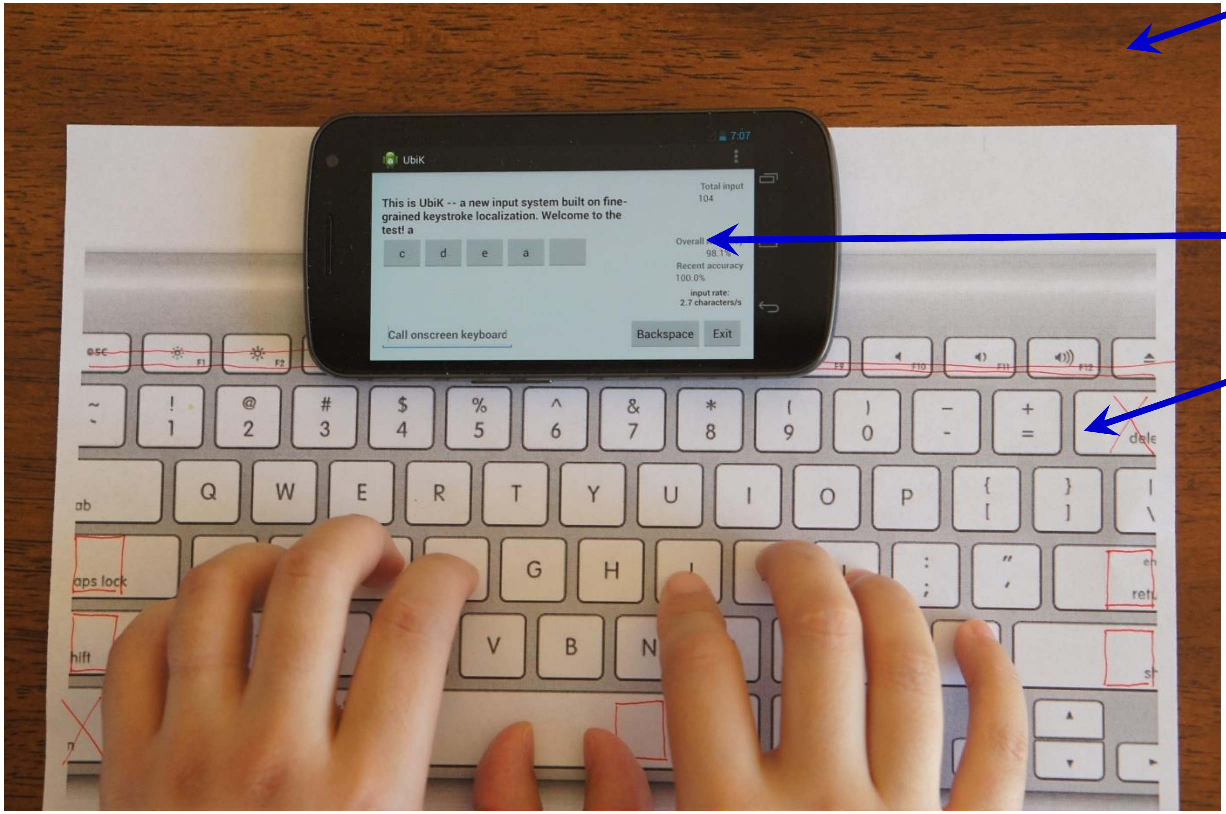


Table (or other flat surface)

Smartphone

Printed keyboard

UbiK: A Short Demo

UbiK: Challenges

#1: Is fine-grained keystroke localization feasible?

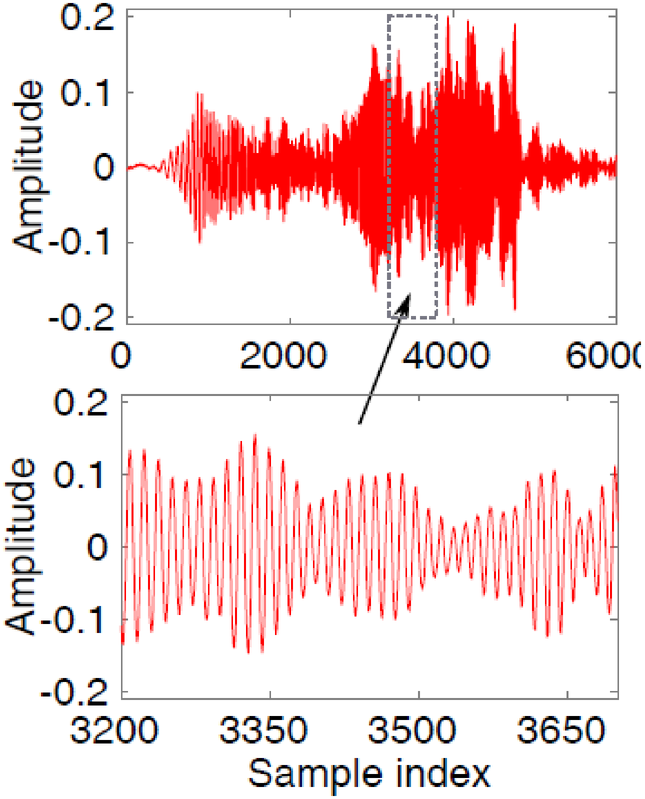
#2: How to make it work on COTS smartphones?

#3: How to make it reliable and robust?

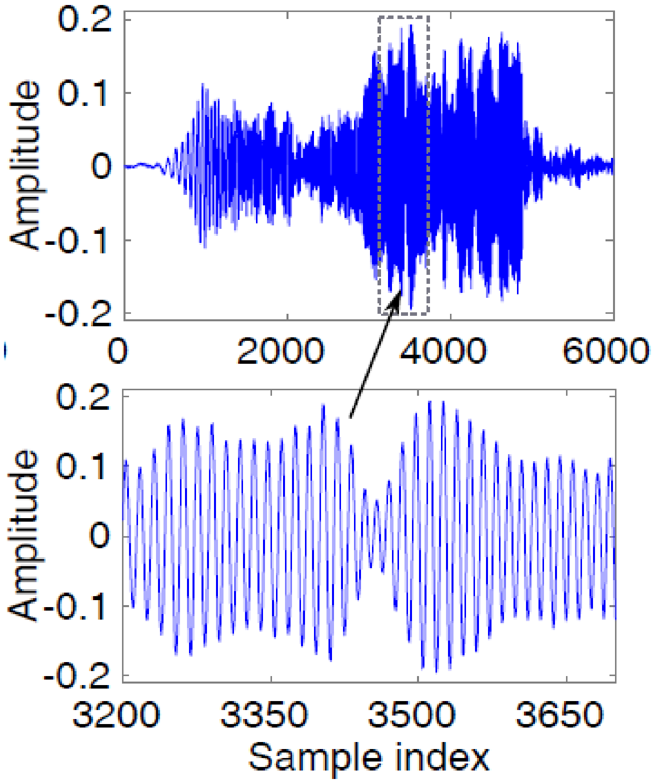
#1: Is fine-grained keystroke localization feasible?

Evidence 1: multipath channel profile as signature

Experiment 1: a chirp tone sent from two different key locations, received by the same microphone



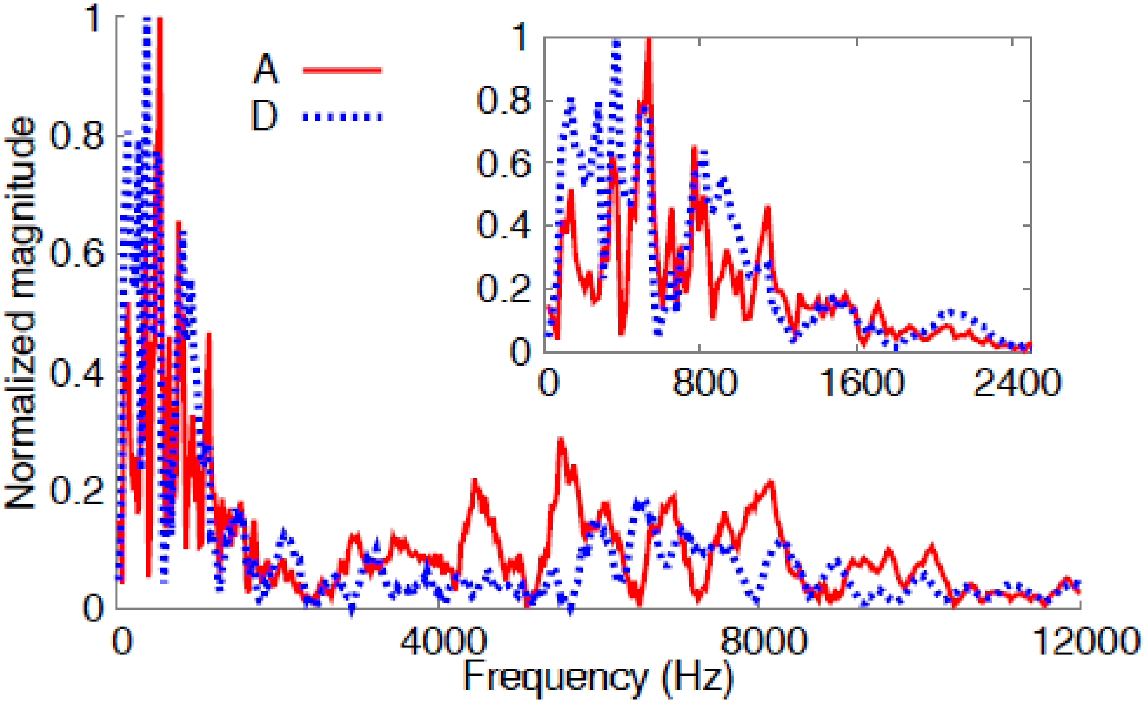
(a) Signals from key location 'A'



(b) Signals from key location 'D'

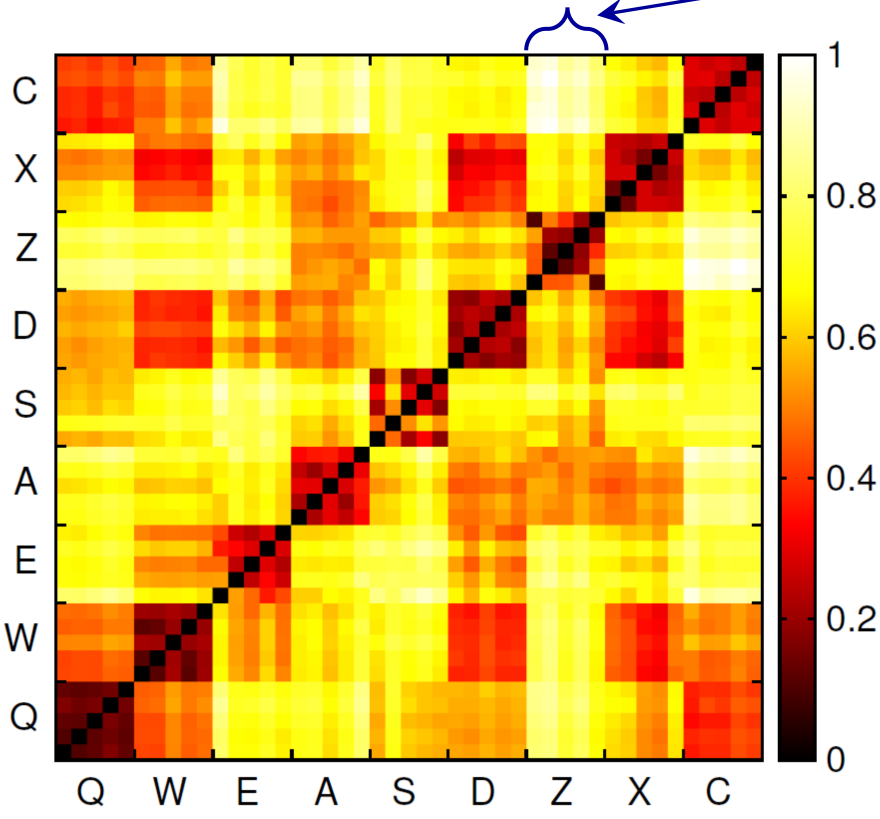
Evidence 1: multipath channel profile as signature

Experiment 2: Amplitude spectrum density of keystrokes at two different key locations

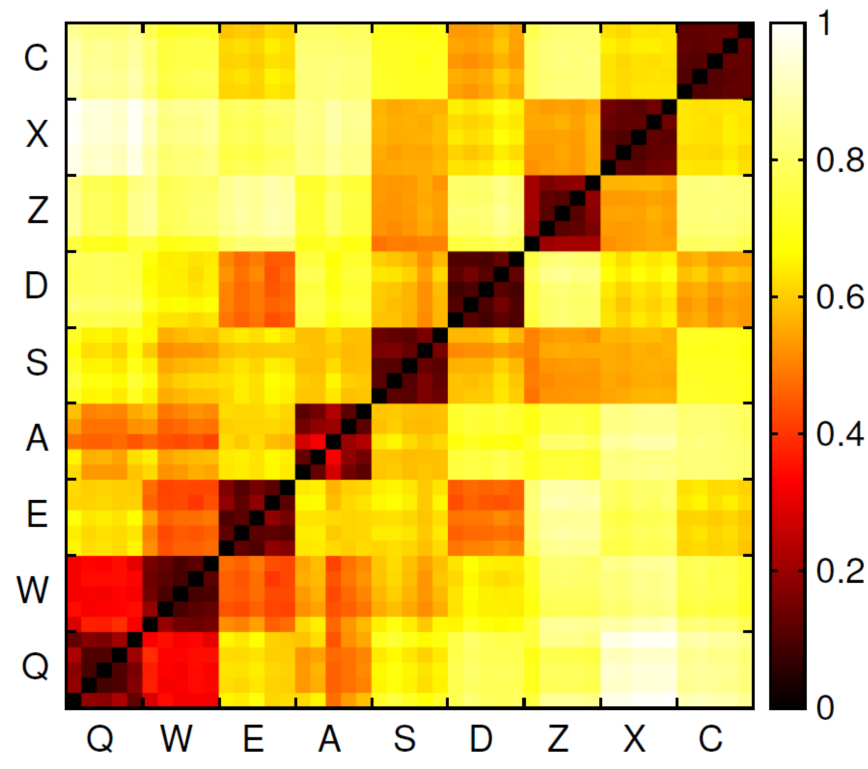


Evidence 2: Spatial Granularity

Experiment: Euclidean distance between ASD of sounds at 9 key locations, each repeated 5 times



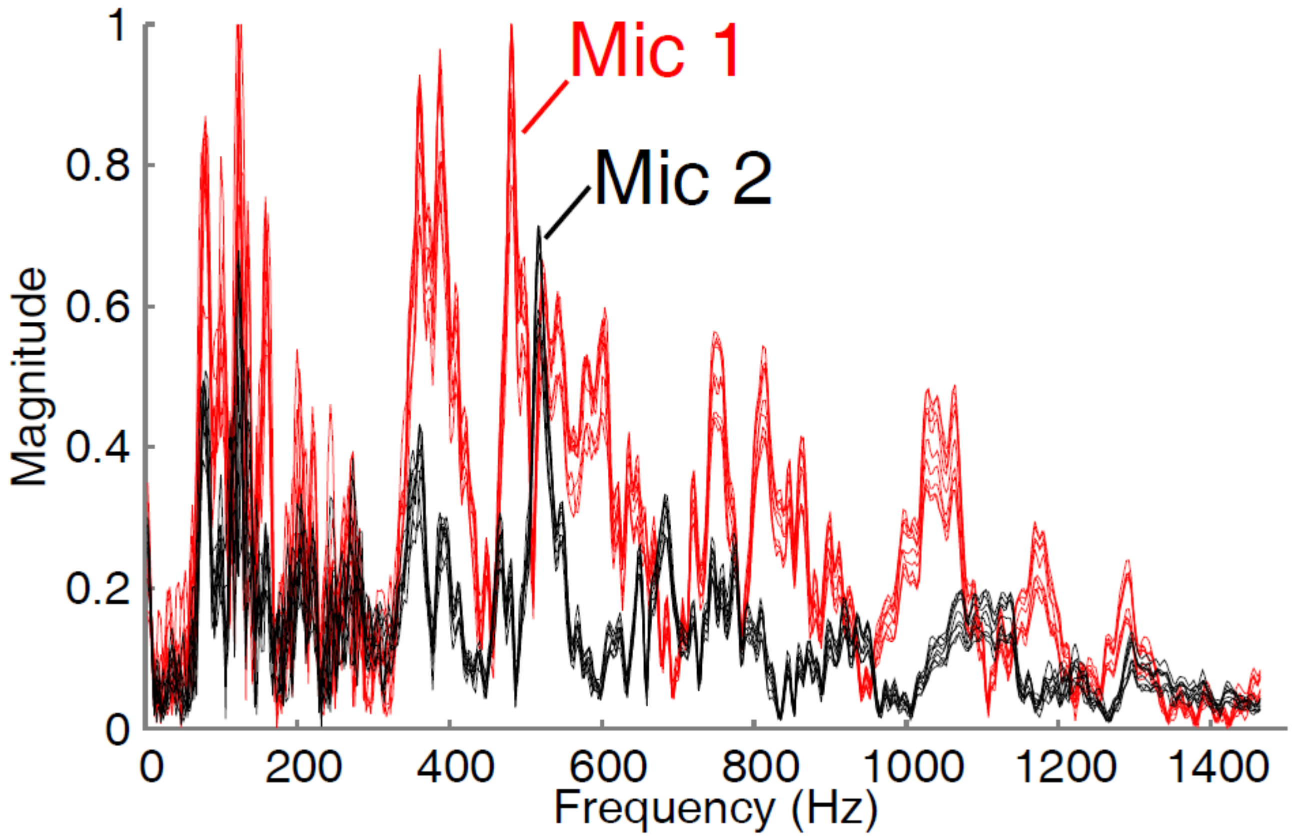
(a) each sound source is created by finger/nail clicking on the key locations;



(b) the sound source is a chirp tone emitted from the key locations.

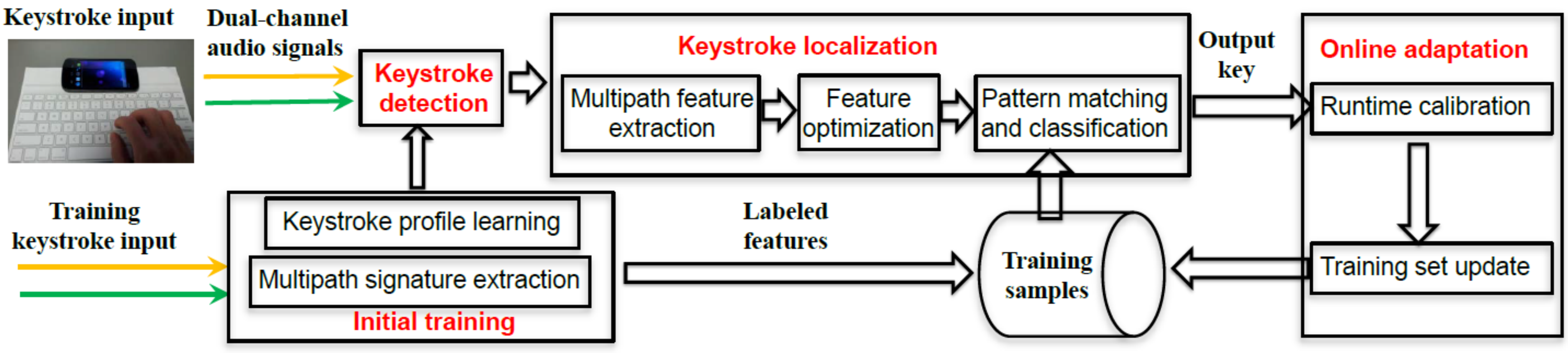
Evidence 3: Diversity from multiple microphones

Experiment: ASD of 10 key presses received by two microphones on the same Smartphone.



#2: How to make it work on COTS smartphones?

Solution Framework



- Core components
  - Detection, localization, adaptation
- Flow of operations
  - Training => typing & adaptation

Keystroke detection

- Basic detection
  - Energy level detection: key detected when sound signal energy passes a threshold (a bit above noise floor)

- How to set the threshold?
  - Adapt threshold using Constant False Alarm Rate (CFAR) algorithm

$$\text{Threshold} = \mu + \gamma\sigma$$

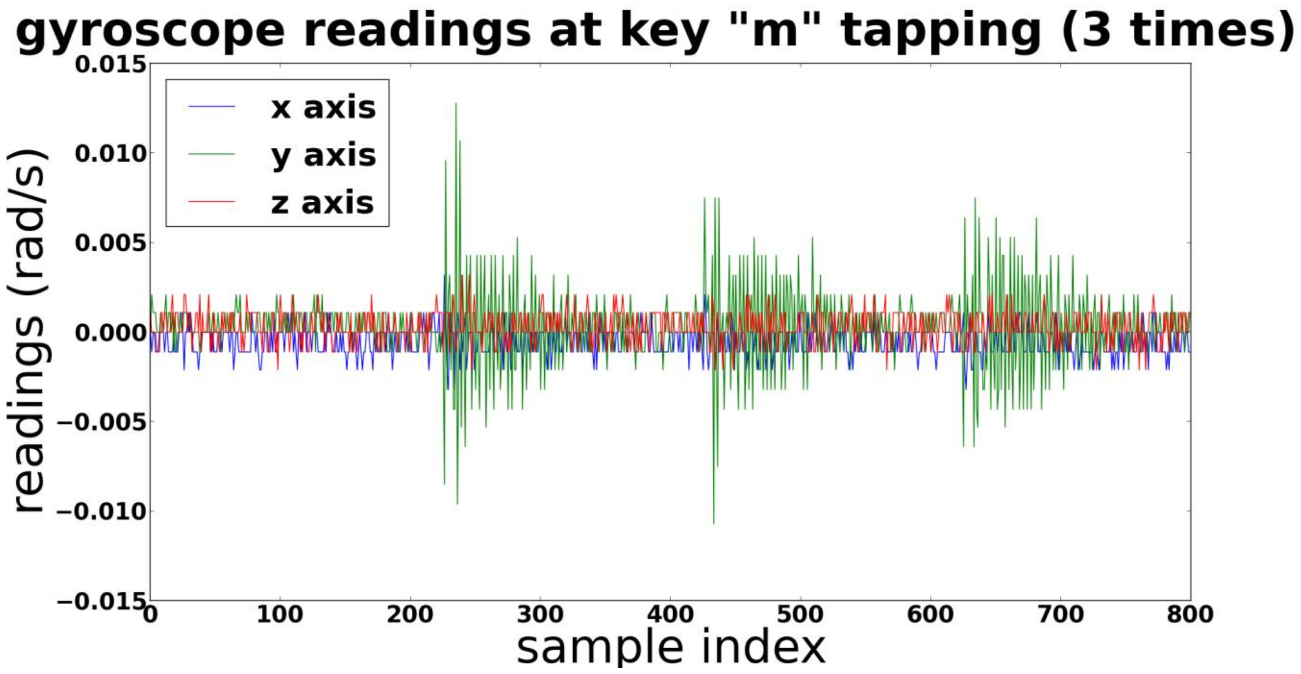
Moving average of noise variance

A scalar (>1) for safe margin

Moving average of noise energy

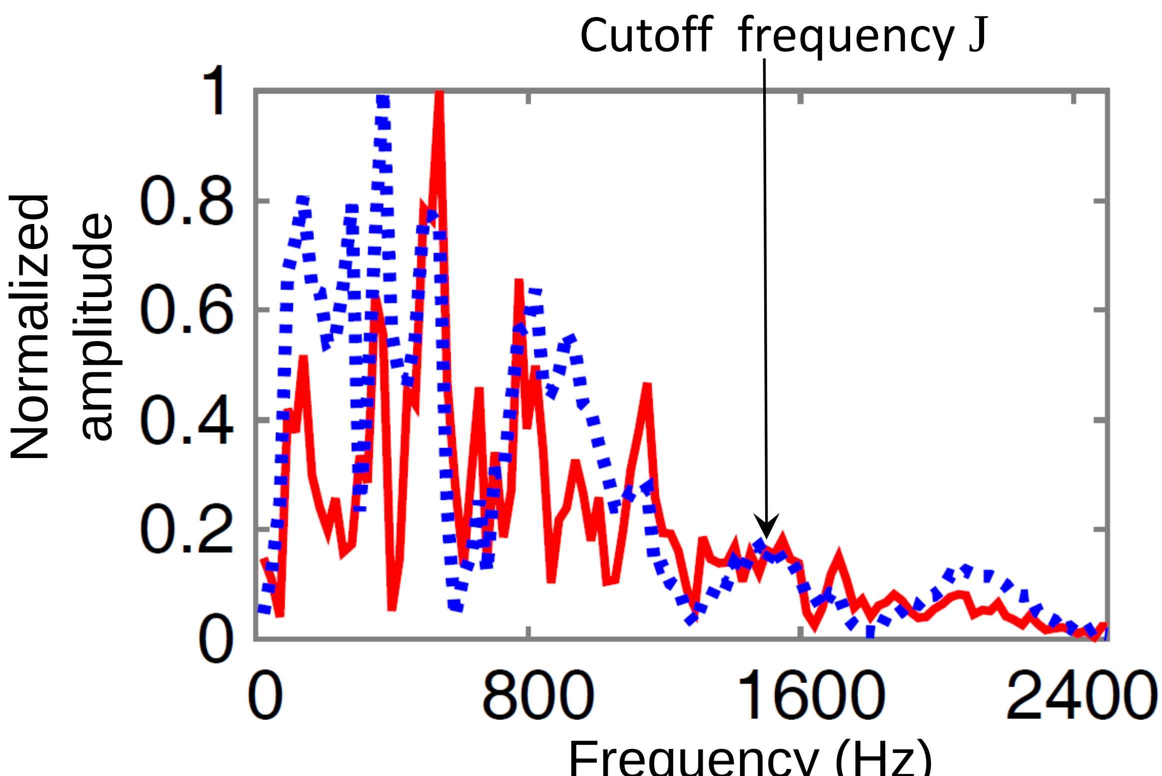
Keystroke detection

- Combating bursty noise
  - Use motion sensor (gyro) to filter out bursty noise



Localization signature design

- Frequency-domain filtering of audio samples
  - High-frequency noise compromise stability of signature
  - Only use ASD below cutoff frequency J in signature
  - J differs on different surfaces



Localization signature design

- Determining cutoff frequency J
  - Our problem: finding optimal cutoff J, assuming all elements in the ASD feature vector have the same weight 1

$$\min_J J^2$$

$$\text{s.t.,}\ y_i\left(\sum_{j=1}^{J} x_{ij} + b\right) \geq 1$$

  - Converted into a feasibility problem: finding the first J that satisfies:

$$b \geq \frac{1}{y_i} - \sum_{j=1}^{J} x_{ij}$$

For all training instances

Localization algorithm

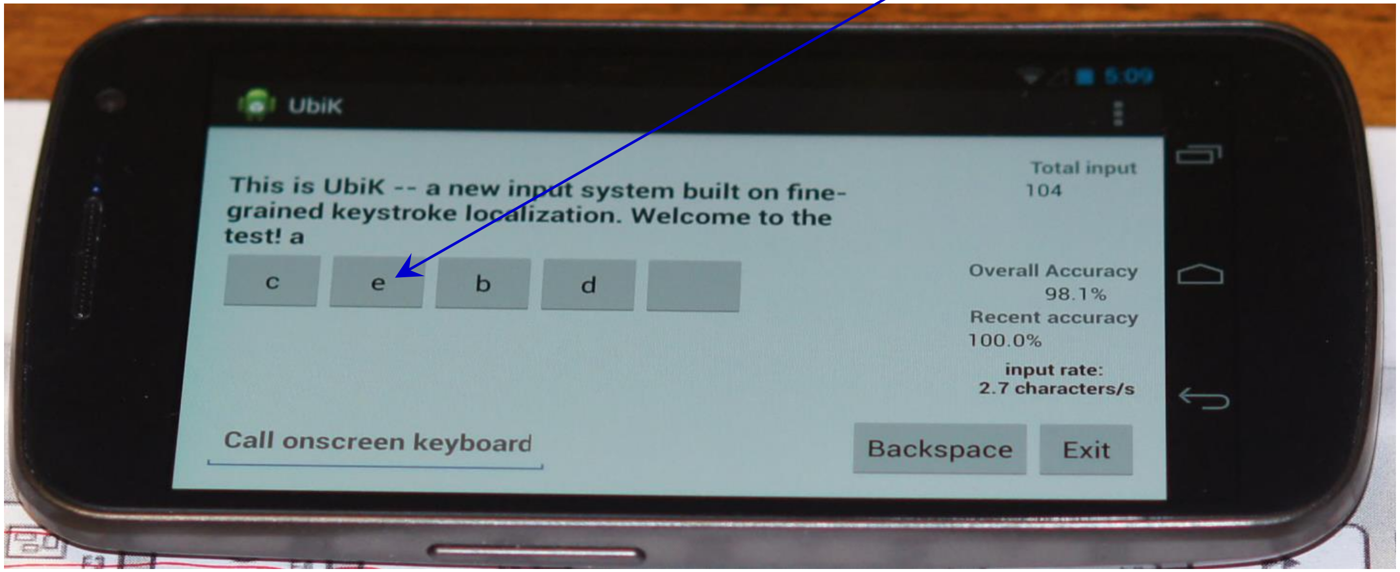
- Simplest pattern matching algorithm
  - Nearest neighbor algorithm
  - Experience: signature design matters more than matching algorithm

- Signal conditioning before matching
  - Normalize ASD
  - Combat variation in click strength

#3: How to make it reliable and robust?

Runtime Adaptation

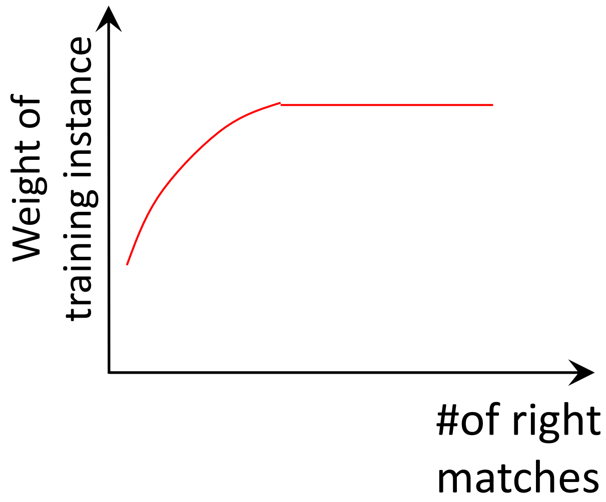
- Runtime feedback provides unique opportunity for improving localization accuracy
  - User's run-time correction, leveraging a candidate list



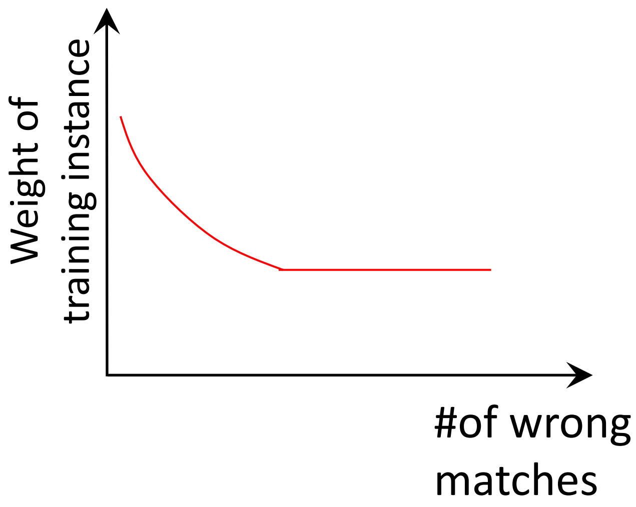
  - Implicit feedback: uncorrected keys => likely located correctly

Runtime Adaptation

- For correctly located keys
  - Insert its ASD into training set
  - Increase weight of training instance that matched it



- For wrongly located keys
  - Decrease weight of training instance that matched it
  - Remove from training set if it ranks lower than set size



Evaluation on Android Phone

Baseline test of accuracy

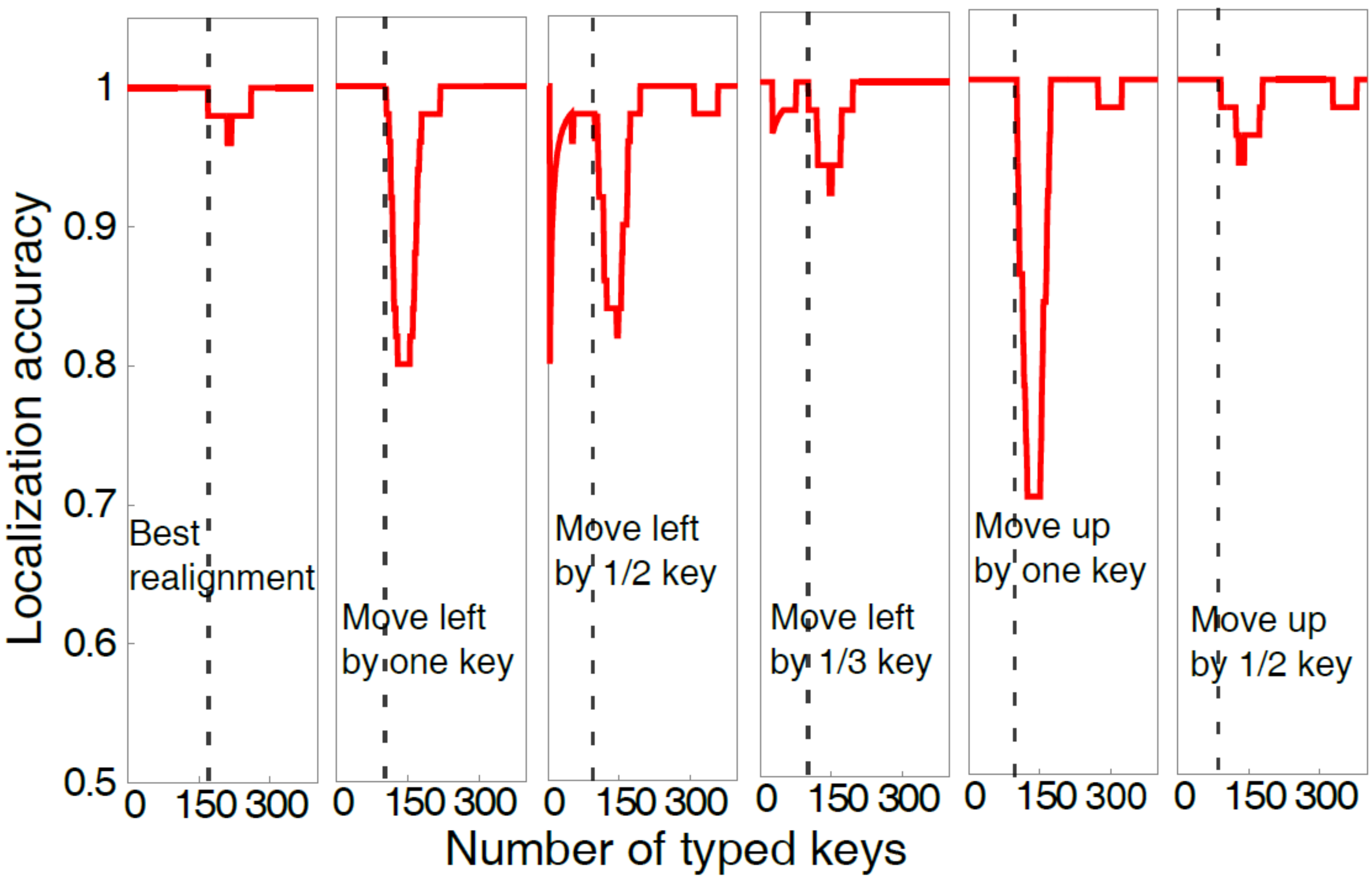
- Detection accuracy

	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%

- Localization accuracy

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

Impact of run-time adaptation



- Quickly recover from disposition of smartphone
- Best practice: reposition phone to original position (Best realignment)

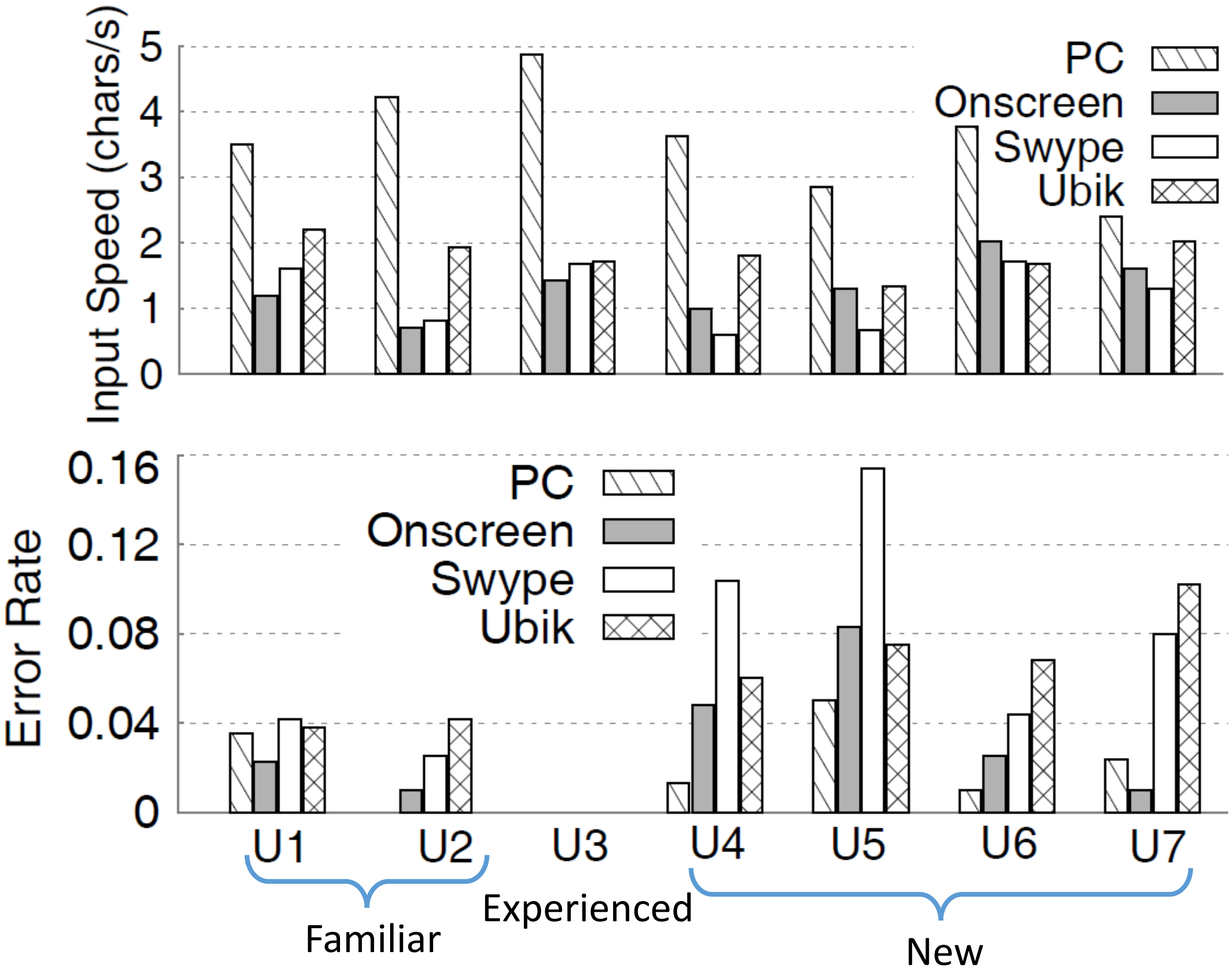
User study

- Setup
  - 7 users, 1 experienced, 2 familiar, 4 new to UbiK
  - Test UbiK in different environment: office, home, library

- Benchmarks
  - PC keyboard, on-screen keyboard, Swype

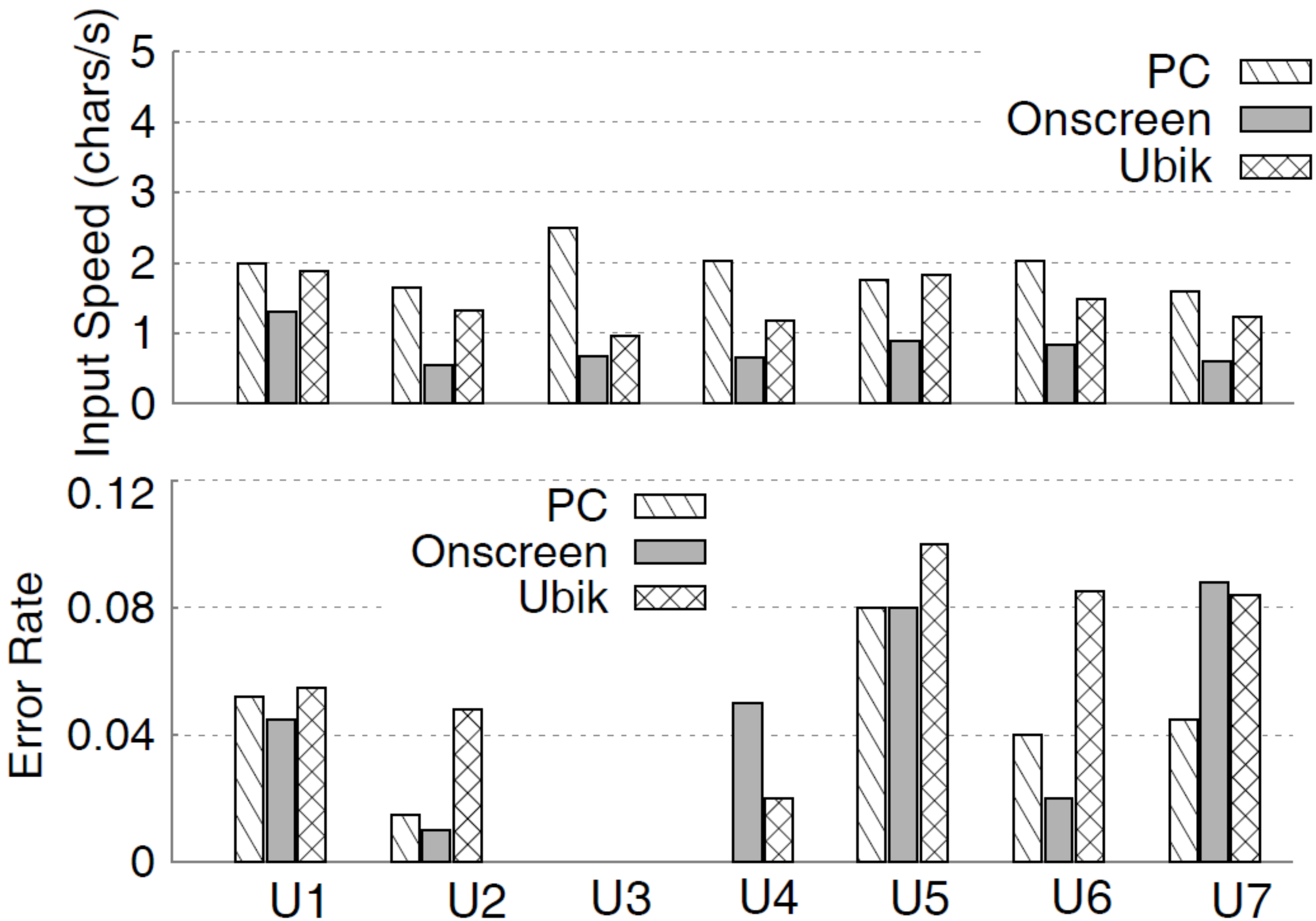
User study

- Real text input



User study

- Random text input



Other concerns

- Lack of kinesthetic feedback
- Hard to type on the correct keys for inexperienced users
- Keystroke sound may be contaminated by noise

Conclusion

- Text-entry: a major problem for mobile devices
- UbiK: cast it as a fine-grained localization problem
  - Amplitude Spectrum Density as Features
  - Detection => localization => run-time adaptation
- Ongoing work
  - Full migration to application-level to support more mobile devices
  - Further enhancement, e.g., dictionary

Thank you!

Welcome to our Ubik demo!

UbiK online demo:
https://www.youtube.com/watch?v=RIIQGNYCFyk&feature=youtu.be

Keystroke detection

- Basic detection
  - Energy level detection: key detected when sound signal energy passes a threshold (a bit above noise floor)

- How to set the threshold?
  - Adapt threshold using Constant False Alarm Rate (CFAR) algorithm

$$\text{Threshold} = \mu + \gamma\sigma$$

Moving average of noise variance

Moving average of noise energy

A scalar (>1) for safe margin

  - Provable performance: false alarm rate is a constant given γ; decreases exponentially with γ