Extending the Capabilities of Smartphone Sensors for Applications in Interaction and Health

Mayank Goel

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

Reading Committee:

Shwetak N. Patel, Chair

James Fogarty

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Extending the Capabilities of Smartphone Sensors for Applications in Interaction and Health

Mayank Goel

Chair of the Supervisory Committee:
Professor Shwetak N. Patel
Computer Science & Engineering and Electrical Engineering

Computing devices continually evolve and in a relatively short time, have morphed from a room-sized machine to something much smaller. Today, a mobile computer can be kept in our pocket, or worn as a wristwatch or a glass. These devices are sufficiently powerful for majority of tasks a typical user performs on a computer. While these devices are slowly catching up with the desktops in raw computation power, they already have much richer sensing capabilities. The on-device sensors provide mobile devices with an unprecedented opportunity to not only provide richer interactions, but also enrich the quality of life.

Acknowledging that the mobile devices cannot be loaded with every possible sensor, I believe a big proportion of sensing responsibility will lie with the generic sensors on our devices, such as camera, microphone, motion sensors, touchscreen, *etc.* In this dissertation, I provide support for my thesis statement:

The generic sensors on mobile devices can be used as substitutes for dedicated sensors in interaction and health applications. In presence of noise and uncertainty, multiple generic sensors can contribute to enable robust and deployable user-facing sensing systems.

In this thesis, I show:

1. How to extend the capabilities of the on-device sensors,
2. Some of the challenges the developer and the user might face while using these generic sensors,

3. Discuss how some of the challenges can be countered by combining multiple generic sensors, and
4. Provide some direction how these sensors should evolve in future.

In terms of application area, there are many domains where the smartphone sensors can prove useful, but this thesis focuses on applications in two domains: *interaction* and *health sensing*.

For teaching me something invaluable, to every teacher in my life.
Especially Gaetano and Shwetak.

Acknowledgments

I would like to thank and acknowledge everyone who has helped me reach this important milestone in my life. I could not have possible done it alone. I got by "with a little help from my friends". This is my way of expressing gratitude to a very small but important subset of people who got me here.

First, I'd like to thank the three most important people in my life: my parents and my wife. Your constant support and love is definitely the most important ingredient in all my achievements.

Second, I want to thank the two people who have been most instrumental in my professional success: Gaetano Borriello and Shwetak Patel. Thank you Gaetano for providing me with all your support and advise when I needed it the most. No words can explain how incredibly lucky I thought I was to be able to receive your guidance. While it saddens me that I won't be able to see you smile as I reach other milestones in life, you will always remain an inspiration for me. Thank you Shwetak for allowing me to be a part of the UbiComp Lab. I still remember how lucky and privileged I used to feel to be part of the awesome group of students you had in the lab. Your creativity, optimism, and energy will always remain inspirational. Gaetano and Shwetak, you are the reason I chose to become a teacher and I would consider myself lucky if I even half as awesome as the two of you!

No amount of words can explain how lucky I am to be a part of the UbiComp Lab. I believe that I was the bridge between the two generations of the lab and I am still mystified by the

never-ending stream of creativity, intellect, and friendliness of this group. I would first like to thank *oldies* of the UbiComp Lab: Eric Larson, Gabe Cohn, Sidhant Gupta, Keyu Chen, and Tien-jui Lee. Eric, my collaborations with you were instrumental in almost everything I talk about in this thesis. I think my Ph.D. would have looked very different had we not collaborated. Your technical support and guidance in my initial years were invaluable and I will continue to look up to you. Gabe, people tell me you are pretty intelligent. May be that's why we never collaborated on anything! Sidhant, you are the reason why I am writing this 300 page document! I don't think I would have chosen to do a Ph.D. without your advice. You were my guinea pig and your awesome experience at UW was an important factor in convincing me to give it a try. This has been the best decision of my life; thank you for convincing me to give it a shot. You and Maneet were my constant source of support; it was invaluable in making sure I had a good foundation in Seattle. Keyu, our paths in the graduate school were almost parallel and I loved working with you. I am going to miss our coffee breaks. They used to be a huge stress-buster for me. Tien, you are a master of MATLAB and signal processing. Thanks for all your help and answering all my questions over the years. Next, I would like to thank the *cool-kids* of the UbiComp Lab: Lilian de Greef, Edward Wang, Alex Mariakakis, Ruth Ravichandran, Elliot Saba, Eric Whitmire (Eric W.), Josh Fromm, Mohit Jain, and Hanchuan Li. I am amazed by your creativity, energy, and friendliness. Make sure you all maintain this strong bond with each other. You all are great and I will cherish my time with you all. I would especially like to thank some of you to whom I have "delegated" a lot of tasks in the past. Lilian, I had a great time working with you. Our working styles are super different. I am sure it was not easy for you at times; thanks for putting up with me. :) Edward, your breadth of knowledge is incredible and I loved interacting and sharing ideas with you. I am sure you will go on to do great things in future. Alex, you are a great researcher and perhaps one of the most friendly people around. I learned a lot from my enthusiasm and am sure I will miss it. Ruth, thanks for all our interactions and discussions. I learned a lot from you (especially when we were working on that heart rate sensing project) and I hope I was able to help you along as well.

I am sure you will be super successful as a researcher. Elliot, I did not initially believe Eric Larson when he said that you are the "Eric 2.0". Not only you are an expert at signal processing, you are also the most selfless person I have met in my life. Eric W., you are the first person who was able to delegate stuff to me. Hats off to you! Thanks for all your help in all things technical, design-related, writing-related, *etc.* I love exchanging ideas with you and I hope we will keep collaborating.

I'd also like to thank Waylon Brunette, Rohit Chaudhri, and Nicki Dell. Waylon, your knowledge of UW's logistics is unparalleled and your support was extremely important in making sure that I settled in quickly. Rohit, I loved working with you and Waylon on ODK Sensors and our interactions played a big role in laying a good foundation for my PhD. Nicki, your continued support throughout my PhD and especially during job search was extremely helpful. I hope we all are able to carry Gaetano's legacy forward and are able to support our students like Gaetano did.

I would also like to thank Jacob Wobbrock and Leah Findlater. You were my advisors on my first research paper and I believe that experience continues to shape me as a researcher. Leah, I am extremely grateful for your advise and patience with me when I had no idea how to conduct research and write a research paper. Jake, you will always remain an inspiration for me. I would consider myself lucky if I inherit even a fraction of your rigor and eye for detail. I would also like to thank James "Bo" Begole. Bo, you were not just a mentor to me during my internship. You were also a friend and I loved interacting with you; and our shared love for Beatles was the perfect icing on the cake. Thanks a lot for your continued advice and support. One person who I cannot thank enough is Gregory Abowd. Gregory, thanks for introducing me to the world of research. I don't think I would have gone down this road had I not worked with you at Georgia Tech. You are the most friendly professor I have ever met and I consider myself extremely lucky to have worked with you, Shwetak, and Gaetano.

Finally, I would like to thank my clinical collaborators for SpiroSmart and BiliCam: Dr. Margaret Rosenfeld, Dr. Jim Stout, Dr. Monsur Habib, Dr. Jim Taylor, and Sharon

McNamara. Your advice and support were instrumental in the success of SpiroSmart and BiliCam. I hope to continue seeking your advice as we continue to contribute to the field of medical devices and health sensing.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

Computing devices continually evolve and, in a relatively short time, have morphed from a room-sized machine to something much smaller. Today, a mobile computer can be kept in our pocket, worn as a wristwatch, or worn as a pair of glasses. These devices are sufficiently powerful for the majority of tasks a typical user performs on a computer. Mobile devices, and especially smartphones, are becoming the preferred computing device for a significant number of users; in fact, a phone is the only computing device for a subset of those users.

While these devices are slowly catching up with conventional computers in raw computing power, they already have much richer sensing capabilities. On-device sensors provide mobile devices with an unprecedented opportunity to not only provide richer interactions, but also enrich the quality of life. These sensors allow phones to be more context-aware and adapt to the user's behavior and preferences. These devices hold the promise to become "invisible", and thus "weave into the fabric of life" [71]. Gregory Abowd, in fact, calls the modern smartphone a "realization of Mark Weiser's predictions" [2].

Perhaps even more importantly, the smartphone also promises to be the next generation health sensor [4, 118, 179], bridge the digital divide, and improve the quality of life in

resource constrained environments [10, 28, 35, 56]. However, many of these intended applications require additional sensor attachments [27, 109, 243]. Having a dedicated new sensor on the device can be quite powerful, but the barrier to entry is also very high. It might not be economically feasible for the hardware manufacturer to build a new sensor into the device for a niche application. The inclusion of new on-device sensors requires "killer apps", *i.e.,* applications that are "essential" and in which "many users can find profit and utility" [6]. A good example of a killer app would be localization. Barometers may seem to be a frivolous sensor to add to a smartphone, but they can improve GPS performance, which increases profit *and* utility for manufacturers, businesses, and users. However, use cases such as detecting newborn jaundice [53] and monitoring the temperature of milk while boiling [35] might not provide enough incentive for manufacturers to include dedicated sensors on a general purpose computing device. As an alternative, the user always has the option of connecting an external sensor. However, that adds an economic and logistical burden on the user. This is especially troublesome for opportunistic sensing [31], social sensing [156], and for technologies for the developing world where the low barrier to entry and seamlessness is highly valued.

1.1 Dissertation Goals

Acknowledging that mobile phones cannot be loaded with every possible sensor, I believe a big proportion of sensing responsibility will lie with the generic sensors on our devices, such as cameras, microphones, motion sensors, touchscreen, *etc.* Throughout this dissertation, I provide support for my thesis statement:

The generic sensors on mobile phones can be used as substitutes for dedicated sensors in interaction and health applications. In the presence of noise and uncertainty, the use of multiple generic sensors can enable robust and deployable user-facing sensing systems.

The main aim of this thesis is to study the feasibility of using on-device generic sensors for various application domains. Depending on the problem, the need for the application's efficiency, accuracy, delay, *etc.* can be very varied. Therefore, it is important

to study the sensors' efficacy in diverse application areas. Interaction and health are two very different areas with very different performance requirements. They also strike a balance between popularity and impact. Every user *interacts* with their mobile phones and therefore any improvement in those interactions can be very impactful. In contrast, not every smartphone user will use the health and diagnostic applications on their mobile phone, but those who need them can find such applications extremely helpful.

From the sensors' perspective, they do not need to work in isolation. Different environments will throw different challenges; and a sensor might benefit from information from other sensors on the phone to adapt to the changing environment, or to simply improve the final inference. To study the opportunities and challenges associated with such an approach, it is important to deploy the applications in noisy, real-world situations. Learning from my experience in developing, deploying, and refining multiple mobile applications in multiple domains, I aim to answer following research questions in this dissertation:

1. How can we substitute dedicated sensors with on-device generic sensors?
2. How do these generic sensors cope with different performance requirements?
3. Can multiple on-device sensors be combined to improve performance?
4. When deployed in noisy, real-world environments, do these sensors cope up and performance reliably?
5. How do these sensors and their infrastructure need to evolve to performance better and cope with noise and uncertainty of use?

1.1.1 Sensors

I propose to use a subset of sensors on modern mobile devices for a host of applications. I specifically focus on:

- Inertial (accelerometer and gyroscope),
- Audio (microphone and speaker),
- Visual (camera), and
- Touch (touchscreen) sensors.

There is no doubt that there will be many applications where these sensors will be insufficient. Still, I believe that these sensors deserve special attention because they are similar to what we, as humans, can sense. It is attractive for hardware manufacturers to keep improving these sensors as they directly drive the user experience on their devices. Using these generic sensors does not come without trade-offs. The very fact that these sensors are generic means that they sense many things and can be extremely noisy. For example, a phone's microphone can be used to sample frequencies above 18 kHz to detect in-air gestures to improve user experience [82]. However, the microphone also listens to many other frequencies and detect all the other motion around it. Those unwanted frequencies can be filtered out quite easily, but the sensitivity to unintended motion is not always straightforward. Overall, it is an interesting trade-off between a newfound capability and a very *noisy enabler*. Although there is no one panacea to these problems, this thesis suggests that we combine information from multiple sources to reduce the sensor "noise". These multiple sources might be: (1) different kind of sensors on the same device, (2) similar sensors on multiple devices, or (3) some external device/object. I will use examples of different works where some of these techniques will help and contend that when deployed in the inherently noisy real-world it is extremely important to understand the noise that the device might experience and iterate through the design. In this thesis, I will also build on my experience of working with these sensors to suggest some directions which these sensors can take to evolve into much more useful and powerful sensors.

1.1.2 Application Domains

Until the last decade, most mobile device-based sensing and HCI research focused on making the devices faster and smarter [233]. These devices were investigated in isolation,

and social and contextual factors were largely ignored. As devices became more intelligent and connected, they also became more social and context-aware. In 2006, Wobbrock [233] mentioned accessibility, context-awareness, education, and healthcare as four areas where smartphones had an unprecedented opportunity to make an impact. Taking motivation from these recommendations, this thesis focuses on applications in the areas of Interaction (with a stress on context awareness) and Health Sensing (Figure. 1.1). Apart from being extremely well-valued by users, these application areas have one more very important characteristic: the cost of a wrong inference can be extremely high. For example, if a restaurant recommendation system suggests a restaurant that the user does not like, the cost for the user will be much less compared to the user's phone making a unintentional phone call in a meeting or showing normal blood glucose level in case of a sugar spike.

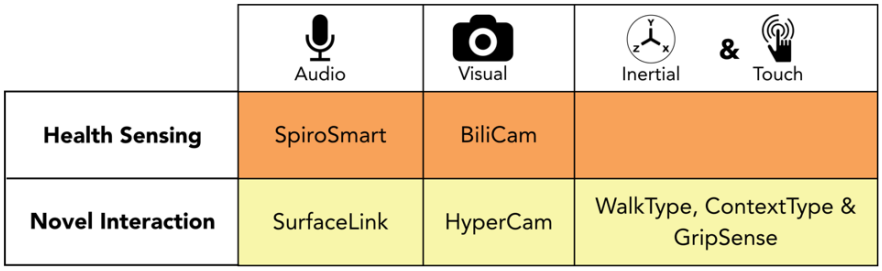

Figure 1.1: This thesis focuses on two application domains: Health Sensing and Novel Interaction.

1.1.2.1 Interaction

A typical mobile user is no longer confined to the static environment of a desk. The user can be outdoors, in motion, or interacting with multiple devices at the same time. Overall, mobile devices are used in a myriad of situations and contexts; however, they are largely unaware of these variations. Unlike the human senses, the sensors on the device are not making the device aware of its surroundings. I believe that mobile devices can use their sensors to adapt to their contexts and also improve the way the user interacts with them in these varied contexts.

1.1.2.1.1 Making the Devices More Context Aware

It is important for mobile devices to develop a better understanding of their surroundings; otherwise, the user's interactions with the device are impeded and lead to a condition called *Situational Impairments* [200]. Situational impairments may be caused by a variety of factors including motion, lighting, awkward posture, *etc.*

In this dissertation, I cover a number of approaches to make mobile devices aware of changes in:

- motion,
- posture,
- user's attention and,
- neighboring devices.

This is, in no way, an exhaustive list of device usage contexts, but it provides a good overview of varied contexts. The techniques described in this dissertation are aimed at providing the reader with a starting point to think about how the device's sensors can be used to make the device more aware, then adapt and improve the user experience.

As a demonstration of how the user experience can improve once a device adapts to the changing posture, Section 3.1 discusses a motion-adaptive keyboard that stabilizes itself as a user walks. Such a technique can reduce the typing errors by up to 50%. These explorations are the first evidence that motion information can significantly reduce typing errors. As another example, the phone's motion sensors can: (1) detect how is the user holding the phone, then (2) if the user is only using the phone with one hand, the motion sensors can improve the expressiveness of the hand by enabling pressure-based gestures.

1.1.2.1.2 Multiple Devices in an Environment

Apart from being used in a myriad of contexts and situations, it is common to have multiple co-located computing devices in the same environment. However, these devices

are largely unaware of one another. The only exception are mobile devices that share the same wireless network. Mobile devices have almost no sense of contextual proximity [98]. For example, mobile devices are unaware of other devices in front of the same user, or the other devices that are on the same surface. In Section 4.1, I will discuss an approach to use the phone's microphone to (1) detect devices that are kept on the same table, (2) detect their arrangement on the table, and (3) sense the gestures performed by the user between these co-located devices. This work has actually inspired more explorations and I have collaborated on other multi-device interaction systems that leverage on-device microphones for in-air gestures [12, 36].

Overall, the approaches discussed in Chapters 3 & 4 serve as a demonstration of the *opportunity* as well as the *challenge* of using the on-device generic sensors to make our devices more aware and thereby (1) improve the user experience, and (2) enable new interaction modalities.

1.1.2.2 Health Sensing

In recent years, health sensing on mobile devices has received considerable attention. This can primarily be attributed to their ubiquity, lowering costs, and computation and sensing capabilities. A key trend emerging from the popularity of mobile devices is the quantified self movement [212]. Devices for fitness, such as FitBit, heart rate monitoring on smartwatches, and other vital signs, are becoming increasingly common. These systems demonstrate how a mobile device, with all its computational capabilities, is an excellent candidate for health and fitness tracking. Barring a few exceptions, however, most of these systems have an external sensor at its core. For example, there are a number of Bluetooth-enabled blood pressure monitors in the market that stream the measurements to a connected smartphone. Although these systems are very useful to keep track of various physiological parameters, they do not improve access to healthcare. Improving access can provide more fine-grained data as the user does not need to carry an additional device. Adding another piece of hardware also makes the technology less accessible to the people in the poorest regions of the world. Providing accurate and

affordable health tracking can improve diagnosis and facilitate control of many diseases. 78% of the world's 7 billion mobile phone subscriptions are in developing countries and almost 10% of those phones are smartphones [1]. Therefore, if we only use the sensors that are already present on mobile devices, we immediately have an opportunity to convert billions of phones into medical devices.

Part 2 of this dissertation discusses medical-grade health sensing systems that use the mobile device's built-in sensors. A significant portion of this part discusses SpiroSmart (Chapter 5). SpiroSmart has the potential to make every single phone in the world (not only smartphones) a spirometer, which can detect abnormalities in a user's lung function. It does this using signal processing and machine learning on top of the audio data recorded by a device's microphone. The audio is then sent to a server for computation and the final results are sent back to the phone. In clinical trials with over 3000 patients, SpiroSmart has performed similarly to FDA-cleared clinical spirometers and within the requirements of ATS guidelines [191]. A significant portion of this dissertation focuses on SpiroSmart to demonstrate how to: (1) prototype a proof-of-concept and easy to deploy sensing application that uses the on-device sensors, (2) deploy the system in real world, (3) learn from the deployments, and (4) iterate over the process to continually improve the system's performance. The dissertation goes on to discuss how this iterative process led to not one solution, but a suite of solutions to solve one problem, *i.e.,* lowering the barrier for lung health assessment. I hope that this part of the dissertation serves as a design guide for creating smartphone-based medical applications.

Using my experience in developing, deploying, and refining mobile sensing systems in noisy, real-world environments, I highlight some of the limitations of the current mobile phones and their sensing infrastructure (Chapter 6). I also recommend how smartphones and other computing devices need to evolve their sensing infrastructure in order to make indirect mobile sensing more potent and uniform across usage scenarios and devices.

[1] The World in 2014: ICT Facts and Figures:
http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2014-e.pdf

Apart from making recommendations on how to improve performance of existing sensors, I also discuss how we can add new capabilities to existing sensors. As demonstrated by most of this dissertations, the on-device sensors are incredibly powerful; but as our reliance on these sensors increases, it becomes important to make sure we add new capabilities to these sensors.

Finally, as the capabilities of on-device sensors improve, the developer will continually face the choice between indirect sensing through on-device sensors and dedicated sensing through off-device sensors. I conclude this dissertation (Chapter 7) with a discussion of various factors that a developer needs to evaluate before making a deciding on their sensing approach.

Chapter 2

Background and Related Work

My research draws motivation from a variety of different areas, including HCI, mobile HCI, systems research around frameworks for internal and external sensors, computer vision, bio-engineering, ubiquitous computing, etc. In this section I focus on the highly related work that was influential in shaping and improving the sensing infrastructure of our modern smartphones.

2.1 Before Smartphones

Smartphone's richer functionality is often linked to its sensors. For example, the accelerometer is used to orient the phone's interface according to user's posture and can also count their steps throughout the day. Before the advent of modern smartphone, many researchers focused on studying the utility of various sensors and how would they improve our lives. These explorations eventually shaped the sensing infrastructure of the modern smartphone. Here, I mainly focus on three seminal works: Mobile Sensing Platform (MSP) by Intel and University of Washington [40, 134], Hinkeley *et al.* 's exploration with various sensors to improve mobile interaction [93, 94], and Dietz and Leigh's DiamondTouch [57] (Figure 2.1).

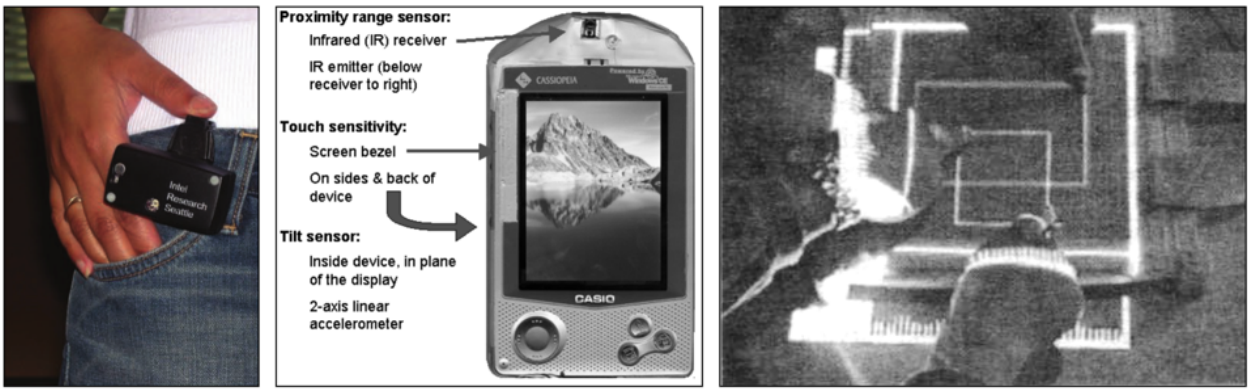

Figure 2.1: Explorations that informed the design of modern smartphones. *(Left)* Multi Sensor Platform, *(Middle)* Prototype by Hinkeley *et al.* , *(Right)* Multi-touch gestures on DiamondTouch.

With the Mobile Sensing Platform, Choudhury *et al.* studied the role of different sensors for localization and activity tracking. They encased different sensors, such as accelerometer, gyroscope, microphone, phototransistor, barometer, IR light, temperature sensor, and compass, in a small box and attached it to the user's waist. They found that an accelerometer, microphone, and barometer are sufficient to locate and infer the activity levels of an individual. And, it is not by accident that today we find all of these sensors in almost all smartphones and these phones reliably detect the user's activity levels.

If we look at the field of improving user interaction with mobile devices, then Hinkeley *et al.* 's [94] investigations with sensors stand out. They attached a proximity sensor, a capacitive touch sensor, and an accelerometer to a mobile device to detect if the user is holding the device next to their ears, or if they are walking with the phone in their hands, or what is the orientation of the phone, etc. They then went on to suggest how should the device react to these situations. For example, changing the layout of the screen according to the device's orientation, or turn off the touchscreen and start recording voice memos when the user puts the device next to their ears. Again, almost all of these sensors and their use-cases made it to the modern smartphones. Citron [243] is another example of a framework that adds sensors to personal mobile devices for improved interaction and activity inference.

DiamondTouch [57] was one of the first explorations into multi-touch surfaces that can be used without any styli. It also suggested using multi-touch gestures to draw bounding boxes around interactive objects. These techniques and gestures have inspired gestures, such as pinch-and-zoom, on modern touchscreen devices.

2.2 Different Types of Sensing on Mobile Devices

Many of the sensors and their proposed use-cases were mainly to drive the device's user interface or figuring out user's environment. However, the power and mobility afforded by the mobile devices has spawned probably the biggest spread of applications in the history of mankind. As Abowd said [2], smartphone is the first truly ubiquitous device and is a realization of Mark Weiser's vision. However, sensors embedded in the devices do not solely drive these advances. A number of other factors, such as better batteries, cheaper and more efficient computation, large-scale application distribution channels (*e.g.,* , Apple App Store, Google Play Store) have played a significant role [129, 156]. By mining data coming in from multiple sensors and from multiple users, it is possible to infer behavioral patterns of not only users but communities [16, 29, 31, 58, 132].

This ease of sensing and deployment of applications has given rise to many different kinds of sensing approaches on mobile devices. Lane *et al.* [129] classify sensing across different dimensions. For example, Personal *vs.* Social Sensing, or Participatory *vs.* Opportunistic Sensing. Personal Sensing is designed to be used and benefited by a individual user. For example, UbiFit Garden [46] tracks a user's activity and encourages them to meet their fitness goals. In contrast, Social Sensing involves collecting data with the help of multiple users and making shared inferences. For example, CitiSense [14, 165] maps pollution data of a city by collecting the data on multiple devices. Such systems only become useful once a large number of people start participating and it becomes a non-trivial proposition if the users have to add sensors to their devices to participate. In case of personal sensing, it is an easier problem because most probably the user is motivated enough to go that extra mile. I claim that for all other applications, it would be much more beneficial to try and leverage

the sensors that are already there on the phones. For example, Ear-Phone [184] uses the on-device microphone to map urban noise pollution.

Another way of classifying mobile sensing applications is by the extent of user's involvement. In Participatory Sensing [29], the user actively engages in data collection; whereas in Opportunistic Sensing [31], the device collects data without any user engagement. It is important to not confuse these with Personal and Social sensing, for example, a personal sensing application can be either participatory (*e.g.,* , [130]) or opportunistic (*e.g.,* , [131]). Both approaches have their pros and cons and need to be studied in detail. For example, the way the system stores and processes the data in opportunistic sensing balances between privacy concerns and computational load. A number of frameworks are designed to enable the developer to find a good balance for their applications. I will discuss these in detail in the next section.

2.3 Mobile Sensing Frameworks

2.3.1 APIs and Frameworks for using Internal Sensors

Before Apple iPhone and Android, smartphones mostly lacked efficient software infrastructure for sensing. The application programming interfaces (APIs) were inadequate to closely control the device's sensors and to manage resources [129]. Recent mobile platforms, including iOS, Android, and Windows Phone, provide better APIs to control the low-level sensors and utilize the computation power of the devices. However, there are still a number of issues; for example, the batteries still do not last more than a day, the sensors can be unpredictable in terms of both accuracy and efficiency, and with devices connected to the Internet, security concerns are also more heightened. Therefore in recent years, a number of frameworks have been developed to solve many such problems.

A number of frameworks have investigated approaches to reduce the power consumption of on-device sensors. For example, Zhuang *et al.* [249] introduced an adaptive

location-sensing framework that improved the energy efficiency of location-based applications. The framework included four design principles: (1) reducing the number of queries to the GPS sensors by picking up the location from the GSM or Wi-Fi radios, wherever possible, as they consume significantly less power than GPS sensor; (2) using less power-intensive sensors, such as accelerometer when the user is static, (3) combining requests from multiple applications in time; and (4) adjusting sensing parameters like frequency according to the battery charge levels.

Traditionally smartphones used their powerful central processors for sampling its various sensors. This approach became expensive as sampling rates increased over 1 kHz. Therefore almost all platforms moved to a more asymmetric architecture with "multiple, memory-incoherent cores that differ in processing power by orders of magnitude" [138]. This way the more frequent but simple task of sampling sensors is off-loaded to lower-power co-processors. Turducken [209] and LittleRock [177] provided a framework that integrated multiple co-processors into a single multi-tiered device. It maintained consistency, ensured always-on availability, and time synchronization. However such asymmetric systems can become difficult to program for the developers. Thus, Reflex [138] is a suite of runtime and compile-time techniques that eased the development by concealing the heterogeneity of the system from the developers. Another framework that makes it easy for the developers to access on-device sensors is Funf[1]. It provides the developers with different alternatives to record sensor data, by varying the developer's involvement. In Funf's most seamless form, the developer uses Funf Android app to configure the sampling requirements and the app automatically logs the data and uploads it to developer's desired DropBox account.

Once the data is logged, the second bottleneck for resources is data processing. The developers have the option of either processing the data directly on the device or sending it to server. The developers can also opt for middle grounds and pre-process the data, send it to the server, and then finalize the processing. This is a critical decision and

[1]http://www.funf.org

depends from application to application. Two chief reasons for processing the data on the phone from transmission or saving are (1) privacy as raw data can be sensitive, especially in case of opportunistic sensing, and (2) computational resources. For example, Larson *et al.* [131] opportunistically record audio data to listen for cough sounds. Later, they transform the data using Principal Component Analysis (PCA) to preserve the privacy of the user before transmitting it to the central server. Additionally there are cases where the computation is too intensive for the mobile device. For example, iOS and Android process the audio data on the server before sending it to their voice assistant tools, *i.e.,* , Siri and Google Now, respectively. SpiroSmart employs a similar approach; this way we can also ensure that all the devices use exact same iteration of algorithms. Shen *et al.* [204] recently proposed to employ the on-board DSP on mobile devices to run the machine learning algorithms. Their results showed that such an approach could reduce energy consumption by approximately 17%.

Another approach to manage computation resources is context-based programming. Kobe [41] considered latency, energy, and accuracy tradeoff for mobile sensing. It generates optimized inference pipeline with multiple classifiers for an application on the basis of the training data entered by the developer. Senergy [110] goes a step further and completely decouples the algorithmic details from the application. It seeks latency, energy, and accuracy priorities from the developer and then Senergy automatically tunes its context-sensing algorithms to meet the constraints.

While Kobe and Senergy are optimized for context-sensing applications, there are some more general-purpose frameworks that tune the machine learning algorithms according to developer requirements. SoundSense [143] is a scalable framework for modeling sound events on mobile phones. It proposes the use of multi-staged, hierarchical classification models that generalize well across applications while recognizing the resource constraints. Auditeur [166] is more general-purpose and developer-centric than SoundSense. It runs as a service on the device and can notify an interested app when a sound event occurs. The developer can use Auditeur to record the sound, annotate it, upload it to a classification server, learn statistical models on the collected data, and then once the model is learned,

perform the classification and notify the app.

2.3.2 APIs and Frameworks for using External Sensors

Apart from utilizing the sensors that are already there on the mobile devices, a number of works focus on adding additional sensors to the device. These sensors can be wired or wireless. However, mostly due to the operating system design, integrating external sensors is not straightforward on a mobile device. Gadgeteer [222] eases development with hardware devices by providing modular hardware and software components. It provides a rapid prototyping platform and simplifies how different hardware components communicate with each other. While Gadgeteer focuses on wired components, Amarino [114] eases connection of Arduino microcontrollers with an Android phone over Bluetooth. It also helps the developers in easily sharing information about phone's internal events such as phone calls, on-device sensor data to an Arduino-powered embedded device over Bluetooth. Another framework to ease connection with wireless sensors in Dandelion [137]. It provides platform-agnostic programming abstraction for sensor data processing. Dandelion envisions a scenario where sensor vendors provide a runtime to enable application-level abstractions. Another popular sensing application middleware is PRISM [51]; it provides reusable components and eliminate redundant efforts to distribute the applications at a large scale. Similar to PRISM, I have collaborated to build ODK Sensors ([27, 34]) that simplifies development of sensor-based mobile applications by enabling all sensors to be accessed through a unified interface. It divides the system into three layers: Application Layer, Framework Layer, and Hardware Layer. It is designed so that the application developer can be completely agnostic to the type of sensor connected to the mobile device; *e.g.,* , whether the data comes from a wired temperature probe or a Bluetooth-enabled, the application code remains the same. The application simply asks for data from the temperature sensor connected to the device and the ODK Sensors framework automatically negotiates connection with the appropriate sensor and shares the data with the app. The main aim of the framework is to decouple the sensor from the app, so that once the application is deployed, the user can switch

between different sensors without changing the application. This capability is particularly attractive when a project is deployed by a remote organization in low-resource setting and there is not guaranteed developer support.

2.4 Applications of Mobile Sensing

2.4.1 Mobile Sensing for Interaction

Mobile sensing for novel interaction is a very popular area and each year several mobile sensing works feature in different academic conferences. In fact, Mobile HCI community almost exclusively focuses on interaction on mobile devices. Here I focus on some of the most highly related work of prominence. Broadly, my exploration in novel interaction can be divided into four areas: situational impairment, text entry, multi-device interaction, and surface gestures.

2.4.1.1 Situational Impairments

It has been emphasized by a number of researchers that the devices need to have knowledge of a user's context or situation to provide better support to the user by making interfaces intelligent and invisible (*e.g.,* [104, 198]). Recent design approaches have also emphasized this; for example, proactively sensing context is a design principle of *ability-based design* [235], which seeks a better match between interfaces and the abilities of the people who use them. There also has been extensive research in the domain of activity recognition to have a better understanding of the context of a user-in-motion. Choudhury *et al.* [40] developed a small wearable device with number of sensors for activity recognition. Laerhoven and Cakmakci [221] leveraged an accelerometer attached to a phone for recognizing different user motions like walking, climbing stairs, *etc.* Schmidt *et al.* [198] leveraged accelerometers to detect, in addition to user movement, whether a device is held in the hand, is on a table, or is in a suitcase. This dissertation contributes to this research area by enabling the phone to recognize if a user

is walking or how is the device held in the user's hand.

Mizobuchi *et al.* [157] evaluated how increasing size of target buttons could improve text entry performance while walking. Lin *et al.* [139] studied the effect of walking on stylus tapping, and found that performance decreased while walking. Yesilada *et al.* [248] demonstrated that the number of errors made by an unimpaired user on a mobile device was similar to a motor-impaired desktop user. Consequently, existing techniques for motor-impaired users may be useful in accommodating situational impairments on mobile devices. Kane *et al.* [107] proposed an auto-correction system to help motor-impaired typists. Mobile devices can also impact the user's ability to read information. For example, walking has been shown to have a negative effect on both text legibility [163] and reading comprehension [17].

To address the foregoing challenges, techniques have been proposed to bridge the gap between stationary and walking interaction. For example, Brewster *et al.* [25] used audio feedback to improve touch screen interaction while standing and walking. Bragdon *et al.* [24] evaluated touch screen gestures in mobile environments, and established that gestures starting on the screen border as a reference point are not "significantly affected by the environment." Taking an adaptive approach, Kane *et al.* [107] coined the term walking user interfaces (WUIs) and evaluated a method to automatically enlarge soft buttons when users are walking versus stationary. Yamabe and Takahashi [244] used accelerometer information to automatically adapt the size of fonts and images while walking. Yatani and Truong [246] investigated how two-handed chorded keyboard input could improve using a stylus-based PDA while walking.

Prior to GripSense, others have also proposed techniques for detecting hand postures. Kim *et al.* [116] and Harrison *et al.* [84] used capacitive touch sensors to differentiate between numerous grips. Taylor and Bove [214] additionally leveraged accelerometers to dynamically detect changes in a user's grip for improved interactions. GripSense, in contrast, requires no additional instrumentation of a modern smartphone to robustly detect handling grips. The main trade-off for this capability is that the user needs to be

interacting with the device for GripSense to make inferences.

Understanding hand posture is important for making devices more intelligent to situational impairments caused by them. Holz Baudisch [99] have evaluated systematic error in target selection due to change in finger posture. Wobbrock *et al.* [237] studied a number of hand postures and evaluated front- and back-of-device finger performance with mobile devices. A number of researchers [97, 112, 113, 230] suggest that although users prefer single-handed operation while using smartphones, traditional mobile interfaces are designed for two-handed operation. Karlson [112] studied such interfaces and evaluated how they impede thumb-based usage. Azenkot and Zhai [13] found that different hand postures induced different touch patterns and affected overall performance while typing on a mobile touchscreen keyboard. AppLens and LaunchTiles [113] attempted to design interfaces mindful of the limited precision and range-of- motion of the thumb. GripSense, using only on-device sensors, robustly detects whether a user is using his left thumb, right thumb, or either index finger to interact with a device.

2.4.1.2 Text Entry

A plethora of touch screen text entry techniques have been developed for both finger and stylus input. Prior work by MacKenzie *et al.* [148] provides an excellent review. Approaches to improve text entry performance with QWERTY keyboard layouts have been proposed, for example, using geometric pattern matching [122] and gestures [121]. Of particular relevance to our work are approaches that combine language model predictions with probabilities from a touch model (*e.g.,* [76, 78, 79]) to improve overall input accuracy. Language model predictions for the next letter to be typed have been used to resize keys, either visibly [5] or invisibly, that is, without showing the adaptation to the user. Gunawardana *et al.* [78] ran a simulation study demonstrating the usefulness of key anchoring when the language model predictions are invisible. That is, regardless of the changing predicted letter probabilities, a center (anchor) area on the visible key always returns that letter, ensuring that a "direct hit" on the key by a user's finger provides a predictable result. WalkType uses a similar approach.

As with WalkType, others have proposed techniques that use models of key-press distributions built on aggregate typing data collected from users. Most commonly, bivariate Gaussian distributions have been used to model individual keys [76, 78]. A small number of projects have introduced models that adapt to individual typing patterns. A simulation study by Rudchenko *et al.* [193] showed that a personalized key-press model using bivariate Gaussian distributions for each key improved performance over an aggregate model. Adapting the location of keys based on the centroid of the user's previous key presses has also been studied for larger devices [73, 92], but no performance benefits have been found in user evaluations.

2.4.1.3 Multi-Device Interaction

Many researchers have proposed techniques for detecting the presence of multiple devices that share some context with each other and enabled different interactions between the devices. Smart-Its-Friends [98] called this concept "context proximity". The authors described how artifacts established connection with each other by detecting a shared phenomenon in their relative contexts. For example, two devices shared data when a user shook both of the devices simultaneously. This technique of detecting synchronous events has been explored by a number of researchers. Ideas such as bumping two devices together [154] or making contact with touch screens at the same time [183] provide a tangible way of initiating connections between two physically disconnected devices. PhoneTouch [199] detects co-occurrence of device vibration and surface touch to facilitate interaction between an interactive surface and a mobile device. Hutama *et al.* [102] used tilt correlations to facilitate interaction between multiple devices. All these techniques have a scalability problem as they either do not work for more than two devices, or they assume a limit of about one user per device. Stitching [95, 96] provides a very interesting way of connecting multiple devices together. The user draws a line with their finger from the touchscreen of the sending device to the touchscreen of the receiving device. This is a very intuitive action for connecting two devices, but it requires both devices to have a touchscreen and is less intuitive for

connecting multiple devices. BlueTable [231] connects multiple devices on a shared surface using vision-based handshaking.

Detecting where the devices are kept in the interaction environment is relatively straightforward in the earlier mentioned vision-based techniques. The main disadvantages of such techniques are their relatively demanding hardware requirements. These technologies cannot be used for impromptu interactions because it is not common to have calibrated cameras set up in the environment. Dearman *et al.* [54] developed a lightweight system where a group of mobile devices used their backside camera to infer their relative orientation. Lucero *et al.* [146] explored how radio tracking technology can be used to determine devices' relative positions, but this approach fails when the devices are densely packed. Similar to SurfaceLink, Kortuem *et al.* [120] used ultrasonic audio signals for relative localization of devices, but they used external hardware and were limited to only detecting whether the device is to the left, to the right, approaching, or moving away from first device. SurfaceLink, in contrast, combines surface gestures and stereo acoustic sensing to infer a multi-device arrangement on a surface in both x and y directions.

2.4.1.4 Detecting Surface Gestures using Microphones

Prior to SurfaceLink, others had proposed techniques for enabling interactions with a surface using acoustic sensing. Paradiso and Checka did some of the early work in this area [171]. They proposed a system that tracked a user's knock on a glass wall by measuring the time delay of arrival (TDOA) of vibrations at 4 different contact piezoelectric pickups. The work by Crevoisier and Polotti [48] was one of the earlier works that explored interaction with day-to-day physical artifacts through sensing the surface vibrations. ScratchInput [85] demonstrated that when a user drags their nails on a textured surface, the resultant vibrations could be used to convert the surface into an ad-hoc interactive surface. SurfaceLink builds on this work to additionally provide single- and multi-touch gestures with varying speed and length. The gestures can be tracked in real-time and because SurfaceLink performs spectral analysis on the acoustic

signal, it performs well even when the signal-to-noise ratio is low. Hence, the user no longer is limited to scratching with just the fingernails. The user can interact using any part of the hand. SurfaceLink is also first to explore how these gestures can be used in a multi-device environment and presents a set of gestures and enabling technologies for effectively interacting among multiple devices.

Seniuk and Blostein [202] explored how the surface vibrations generated by pens on a known textured surface could be used for recognizing a fixed dictionary of 26 words written by the user. Murray-Smith *et al.* [162] also used such surface vibrations to develop a hand-held interaction device that could be controlled by tapping, scratching, or rubbing the surface. In a similar effort, Harrison *et al.* [87] demonstrated how structured patterns of physical notches could be used as acoustic barcodes when swiped with a hard object like a fingernail or a smartphone. TapSense [86] and similar work by Lopes *et al.* [140] leveraged the contact microphone on a touch screen to detect different touch modes while a user tapped on the screen. SurfaceLink applies this same approach, and further extends the gesture set. To the best of our knowledge, SurfaceLink is the first work to demonstrate and evaluate gesture length, speed, direction, multi-touch, and application of these gestures in a multi-device environment.

2.4.2 Mobile Sensing for Health

With its growing ubiquity, power, and on-device sensors, smartphone is increasingly becoming an attractive platform for medical and health applications. Klasnja and Pratt [118] provide an excellent survey of space of mobile-phone health interventions. They discuss how the capabilities of modern smartphones make them suitable for health interventions and one of the highlighted capabilities is connectivity. While sensors and computation power is important to infer the user's state, connectivity is paramount for providing the intervention. A number of smartphone health applications can connect, over a wired or wireless connection, to devices to measure the user's physiological data, such as their weight, blood pressure, heart rate, glucose levels, *etc.*

Another important facet in health sensing is the relationship that people share with their phones. Phones are often deeply personalized, remain in close proximity of the user, and are used for varied activities throughout the day, from calendaring to email to games. More recently, Agu *et al.* [4] presented a survey of medical sensing solutions using mobile devices and examined potential benefits and challenges. They claim that many mobile health applications involve three steps: (1) gather raw sensor data, (2) process data using machine learning or some other inference method, and (3) display results. This is precisely the methodology I follow in the health sensing solutions proposed in this thesis. However, not all solutions use machine learning or inference to provide interventions. For example, DietSense [187] provides the user with capability to take photo of their food and then efficiently share it with their group or a medical professionals. It is an excellent example of how, in some cases, inference is not important to get the right intervention.

There are a number of such journaling applications in literature. For example, UbiFit [46] is an application for encouraging physical activity. It tracks the user's cardiovascular activity and the user can view their progress through different incentive graphics. Similarly, Wellness Diary [153] is a journaling application that logs various health-related states, such as food intake, weight, mood, stress, blood pressure, amount of sleep, *etc.* A number of studies have shown that such self-monitoring applications can positively impact the user's health and fitness levels [118]. Considering a number of these applications benefit from a group of users sharing their progress with each other, lowering the burden of using such applications will be immensely helpful. One way of lowering the usage burden would be to efficiently use sensors to help the user in seamlessly inferring their activities and physiology.

2.4.2.1 Sensing Physical Activities that Affect Health

Sensing and understanding the effects of exercise and physical training are very popular areas of research. It stretches all the way from pedometers [45][2] [3] to developing systems that motivates the user in increasing their physical effort [61, 160, 218]. Apart from exercise, another activity that directly affects the user's health is eating. It has been very popular area of research in HCI and Ubiquitous Computing and a number of efforts have focused on automatically inferring the user's eating behavior. BodyBeat [178] uses non-speech body sounds, such as sounds of food intake, laughter, cough, *etc.* to infer the user's behavior, activity, affect, and physiology. Apart from BodyBeat, many efforts have focused on detecting the user's eating activity using a microphone. Using an in-ear microphone, researchers have shown that one can detect when (and sometimes what) a person is eating [9, 167]. More recently, researchers have put proximity sensors in the ear to observe similar eating events and detect mastication [19]. In an alternative approach, Thomaz *et al.* [216, 217] explored the use of wrist-mounted inertial sensors to detect and classify eating events. Such efforts can go a long way in helping users stay motivated in controlling their diet and sharing their behavior with each other.

2.4.2.2 Physiological Monitoring and Disease Diagnostics

In addition to physical activity, many efforts have used mobile devices and sensors for direct physiological monitoring. Wello - a proposed specialized smartphone case embedded with various sensors, promises to let people measure their heart rate, temperature, blood pressure, pulse oximetry, and lung function from their phone. Franko *et al.* [70] use a smartphone and a custom plastic accessory for screening patients with scoliosis. Dell *et al.* [55, 56] and Shen *et al.* [206] have also demonstrate the use of smartphones for point-of-care diagnostics, which require visually analyzing test results from blood or urine samples on specialized materials. A number of researchers have also

[2] www.fitbit.com

[3] jawbone.com/up

studied how external sensors can be connected to the phones for health sensing [27, 34, 137, 150]. Using these sensors, smartphones have also started to emulate standard medical devices. For example, Mobisante[4] develops hardware for portable and affordable ultrasound examination.

Many efforts in the area of physiological monitoring and disease diagnostics leverage the on-device sensors. For such efforts, the on-device camera has proven to be an extremely capable sensor.

2.4.2.2.1 Camera As A Health Sensor

Researchers have investigated assisting rehabilitative physical therapy using a depth camera [33] or with infrared cameras in a touch-screen table [21]. Pamplona *et al.* [170] demonstrated a method to screen eyes for specific impairments using an instrumented smartphone camera. Similarly, Bourouis *et al.* [23] developed a camera-based system to detect skin cancer. Researchers have also used the smartphone to detect retinal cancer [225] and tracking foot ulcers from diabetes [228]. BiliCam [53] measures a newborn's bilirubin levels using the smartphone's camera. Bilirubin is a chemical compound in the blood that is produced by the breakdown of old blood cells. Extreme bilirubin levels can be dangerous and need to be treated promptly. The condition causes yellow discoloration of the skin and BiliCam uses the phone's camera to track such discoloration. Similar to BiliCam, Eyenaemia[5] uses a color calibration card placed next to the patient's eyelid and measures the redness of the underside of the eyelid. This redness is affected by the body's hemoglobin concentration. In contrast, HemaApp [227] measures the concentration of hemoglobin in blood at the fingertip and estimates actual concentration of hemoglobin in patient's bloodstream. In the next few paragraphs in this section, I focus on discussing some recent efforts to seamlessly monitor a user's cardiac and pulmonary health.

---

[4] http://www.mobisante.com

[5] www.eyenaemia.com

2.4.2.2.2 Cardiac Health Monitoring

Grimaldi *et al.* [77] use the smartphone's camera and LED flashlight to measure pulse from the fingertip using photoplethysmography. A number of smartphone apps, inspired by this and other similar efforts [164], are now available on various application stores on the Internet. While these applications require the user to be in contact of the phone, Poh *et al.* [175] use a tablet's camera and blind source separation of color channels to measure pulse at a distance. Wu *et al.* [239] invented a technique that magnifies small and slow-occuring changes in a video. This technique has since been used to detect a user's heart-rate at a distance by capturing the micro-flushes on the user's face from blood flow. Moving beyond cameras, Olmez *et al.* [168] and Neuman *et al.* [164] use microphones to extract heart rate using a mobile phone's microphone. Some systems use high-end microphones to detect certain audible manifestations of high blood pressure referred to as *Korotkoff sounds* [7].

More recently, researchers have started using radio frequencies (typically above 2.4 GHz) to detect a user's pulse at a distance. Adib *et al.* [3] and Rahman *et al.* [180] leverage small radio modules to monitor the modulation of transmitted RF waves as they stike the user's moving chest. Considering these systems are essentially monitoring chest movement, apart from heart rate, they can also monitor the user's respiration rate.

2.4.2.2.3 Pulmonary Monitoring

Adib *et al.* [3] and Rahman *et al.* [180] monitor the user's respiration using small radio modules. WiBreathe [185] also demonstrates similar capabilities and discusses the effect of multiple users and noise in the environment on the system's performance. Apart from providing information about a user's respiration rate, such systems are extremely useful in assessing the user's sleep quality. Traditionally, it is measured wither using actigraphy through body-worn sensors [8] or bedside sensors [123]. A commercial product, Zeo[6], uses electrical activity of the brain to estimate sleep quality. One of the most commonly

[6]http://www.digifit.com/zeo/

detected condition while tracking sleep quality is sleep apnea. ApneaApp [181] uses the smartphone's microphone to continuously track a user's respiration and detect sleep apnea events. A number of other efforts (including SpiroSmart - which is discussed in detail in Chapter 5) use the phone's microphone to detect pulmonary conditions. Listen-to-nose [37] and SymDetector [211] use the microphone to continually monitor nasal conditions, such as sneezing, sniffle, cough, and runny nose. Wheeze detection with in-air and throat microphones has also shown promising results in diagnosing the severity of lung impairments, such as asthma [100]. StressSense [142] also leverages the microphone to continuously monitor the user's speech and detect their stress levels. Such systems require to record the audio continuously and the user's privacy becomes a very important concern. Larson *et al.* [131] invented a technique that uses the on-device microphone to detect the number and severity of cough episodes. They used Principal Component Analysis (PCA) to convert the user's speech into gibberish while retaining the legibility of the cough sounds.

The diversity of efforts and explorations covered in this chapter indicates the popularity of sensors (and especially on-device sensors) for solving problems in various domains. As we progress towards a more sensor-rich and connected future, the use of these sensors is only going to increase. This thesis derives inspiration and progresses the work of past efforts in the areas of interaction and health sensing.

Chapter 3

Situational Impairments

The typical "computer user" is no longer confined to a consistent and comfortable environment. A typical user is holding a device roughly of the same size as their hand, is perhaps outdoors, perhaps in the rain, or perhaps even in motion. The traditional assumptions about the the user's environment no longer apply. While the user is aware of these changing environment, their devices are not. Unlike human senses, the sensors on mobile devices do not provide the device with a strong situational awareness. Interacting with these devices can be particularly challenging in these dynamic situations, a state that can be thought of as causing *situational impairments* [200]. This dissertation proposes to use the device's on-board sensors to make the devices more aware of their usage environment. I use the phone's motion sensors, touchscreen, and camera to detect and adapt to various contexts in which the device is being used. In this chapter, I focus on discussing how to counter for situational impairments caused due to the user's state of motion (Section 3.1), hand posture (Section 3.2), and attention levels (Section 3.3).

3.1 Motion

Interacting with the mobile devices is particularly challenging when the user is in motion [139, 147, 157], a state that can be thought of as causing situational impairments.

These situational impairments are exacerbated for mobile text entry on virtual keyboards because of the many repeated targeting actions that take place in quick succession. Researchers have explored various techniques to accommodate some situational impairments, like walking versus stationary interaction [157, 197] and, to a lesser extent, adaptive techniques to automatically meet such needs [108, 244]. Despite these advances, techniques to improve interaction in the presence of situational impairments are relatively unexplored, particularly for text entry.

In this section, I discuss *WalkType*, a smartphone keyboard that uses the device's built-in accelerometer to improve text entry performance while the user is walking. Taking inspiration from image stabilization techniques of camera to remove motion blur, WalkType compensates for imprecise input by incorporating multiple features computed from the accelerometer data: displacement, acceleration, and inference about the user's movement. Additionally, WalkType uses tap location and the finger travel distance during taps to improve the user's text entry performance. Previous work on adaptive text entry has focussed on adjusting key-press probabilities based on language models [5, 76, 78, 79] and touch models [13, 64]. WalkType, to my knowledge, is the first work that has demonstrates that motion information from acceleromter can also contribute to a smartphone keyboard's performance.

In an evaluation with 16 participants, WalkType improved typing performance for both sitting and walking, but the benefits are greatest for walking. WalkType improved text entry speed compared to a control condition from 28.3 to 31.1 words per minute (WPM). Uncorrected error rate [210] also improved, particularly for walking, where average error rate dropped from 10.5% to 5.8% with WalkType. WalkType was also highly preferred by participants, who recognized its performance benefits despite there being no visual difference between WalkType and the control keyboard.

3.1.1 System Design

WalkType uses multiple sources of information to classify a user's finger-touches as key-presses. Among these sources is data from the device's built-in accelerometer, which is used to account for extraneous movement while the user is walking. In this section, I outline WalkType and the process taken to build it, including a study to collect training data from 16 participants. In the next section, we describe a controlled study of the final WalkType system.

3.1.1.1 Building Models

Along with accelerometer data, WalkType uses tap locations and tap travel distance to better predict the intended key. The Weka machine learning toolkit was used to generate two J4.8 Decision Tree models with pruning confidence set to Weka's default (0.25). For classification, the first model used time-domain accelerometer data between taps and the second model used the pattern of accelerometer data generated from the three axes due to the phone's motion while walking. The final WalkType system combined output from both of these models along with a simple Euclidian model. The analysis (discussed later) showed that this composite model performed better than individual models.

For clarity, the term *Euclidian model* refers to a simple key-press classification model that takes as input the *(x, y)* coordinate of a finger-touch and returns the letter whose corresponding key's visual bounds contain those coordinates. The models were built based on typing data collected from 16 participants (10 males, 6 females) who each volunteered for a 45-minute study session. All participants self-rated as expert computer users and intermediate to expert touch screen smartphone users. They were between 21 and 35 years of age (M = 28.69, SD = 4.48).

I built a custom data collection application, WalkType Collect, for the Apple iPhone 3GS that records the device's movement using the on-device low-noise tri-axis accelerometer. In order to elicit natural typing patterns, I followed the approaches of Gunawardana *et al.* [78]

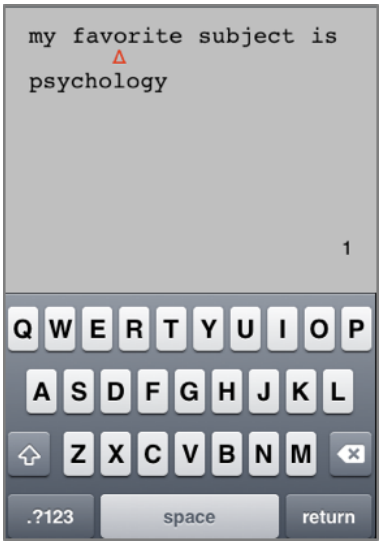

Figure 3.1: In *WalkType Collect*, the user was only given feedback about whether they pressed a key or not. The red triangular cursor moved forward after every key-press.

and Findlater *et al.* [65] and created Collect's keyboard in such a way that it only gave the user feedback that a tap had occurred, but not where it occurred or what key had been hit. To convey this feedback, a small cursor moved under the phrase as the user typed. Figure 3.1 shows an example. If the user realized that they were off by a character or two while typing, they could swipe from right to left anywhere on the screen to delete one tap at a time. The participants were requested not to go back through the whole phrase in order to correct a supposed error. Participants were asked to enter 50 phrases in 2 postures, *sitting* and *walking*, while holding the device with both hands and typing with both thumbs. The order of postures was counterbalanced and participants were randomly assigned to orders. Short phrases of English text from MacKenzie and Soukoreff's phrase set [149] were used. Apart from these, every fifth phrase was a randomly selected pangram from a list of 35 pangrams to ensure sufficient data for all letters of the alphabet.

The lack of tap-location feedback meant that users made mistakes while entering text, which added noise to our data. Thus, outliers were removed during post-processing by eliminating all taps that landed outside the Euclidean bounds of the intended key or its immediate neighbors. About 2.5% taps were filtered out in this process.

3.1.1.2 Displacement and Acceleration Model

One of the major reasons for inaccuracy in typing while walking is the general movement of the phone and its displacement from a relatively stable location with respect to the user. Based on this hypothesis, the Displacement and Acceleration Model improves tap accuracy by incorporating acceleration features in all three axes, and magnitude and direction of displacement in the *z*-axis. To calculate these features, the data from the smartphone's on-device accelerometer was first passed through a low-pass filter to remove noise. This model also includes the base set of features.

To calculate the acceleration features, the filtered accelerometer data was resampled to 10 samples between two consecutive taps. This sampling rate was selected to maintain a balance between reasonable resolution and did not overly increase the number of features for the classifier. These 10 samples of *(x, y, z)* values constitute 30 features for the model. When dealing with accelerometer data, it is often necessary to compensate for gravitational pull on the three axes. For Walktype this compensation is unnecessary because phone orientation stays relatively constant while typing.

For the displacement magnitude and direction features in the *z*-axis, I first subtracted the mean acceleration from the filtered data and then double-integrated it to obtain a coarse displacement estimate. The direction in which the phone moved in the *z*-axis was also calculated by comparing the device's instantaneous acceleration with the moving mean acceleration of the device. If the instantaneous acceleration was less than the mean, the device was inferred to be moving forward. Otherwise, it was moving backward.

I conducted a 10-fold cross-validation on the WalkType Collect data to evaluate the Displacement and Acceleration Model. The model improved classification accuracy on average from 72.8% (for the Euclidian model) to 94.6%. This is a significant increase in overall accuracy (paired-samples *t*-test: $t_{15} = 22.23$, $p < .001$). To evaluate the benefit of the accelerometer data, I also tested this model after removing all accelerometer features. Classification accuracy dropped to 90.8% on average, which was a significant decrease

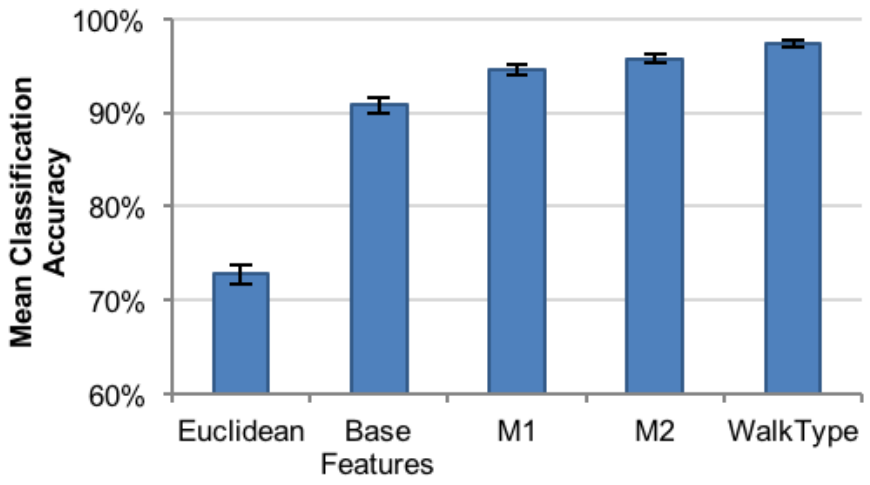

Figure 3.2: Classification accuracy with different models. M1 is the Displacement and Acceleration Model and M2 is the Walking Pattern Model. The difference in performance of Base Features and WalkType illustrates the benefit of the accelerometer data. Error bars show standard error.

(paired-samples *t*-test: $t_{15} = 12.95$, $p < .001$). See Figure 3.2 for a comparison of all models, including the Displacement and Acceleration Model.

3.1.1.3 Walking Pattern Model

When a user walks and types on their phone, the device oscillates in a largely repeatable pattern. Figure 3.3 shows one such instance of the pattern from the accelerometer's *x*-axis. The Walktype Pattern Model leverages the on-device accelerometer to obtain this pattern in all three axes. In addition to the base set of classification features, it incorporates four new features per axis. Crossan *et al.* [49] observed a similar pattern and analyzed how it could be used to detect phases where the user is more comfortable performing target selection with a stylus. WalkType uses slightly different techniques as a user's interaction with a device while typing is very different while walking.

To make the model adaptive to different walking speeds, I calculated the dominant frequency of the user's motion and its mean amplitude from all three axes. This gives a proxy for detecting changes in the user's speed and intensity of movement. WalkType

uses Fast Fourier Transform (FFT) to calculate the signal's dominant frequency. This frequency and its amplitude constitute the first two features. For the third feature, the direction of the last mean crossing before the current tap gives a measure of the direction in which the device is moving. Finally, to pinpoint where in the pattern a tap event occurs, we use the elapsed time since the accelerometer signal crossed the mean value of the signal, as demonstrated by T1 and T2 in Figure 3.3.

These features in the *x*-axis are particularly useful in detecting the user's footstep pattern. I observed that when users' feet hit the ground, their taps tended to shift slightly towards the center of the keyboard. I also observed that a shift to the left was more common when the left foot hit the ground, and a shift to the right was more common when the right foot hit (Figure 3.4). Hence, WalkType can use the data from the accelerometer's *x*-axis to detect which foot strikes the ground. If the low-pass filtered *x*-axis data (red waveform in Figure 3.3) is less than the mean, then the user's left foot has landed, and vice-versa for the right foot.

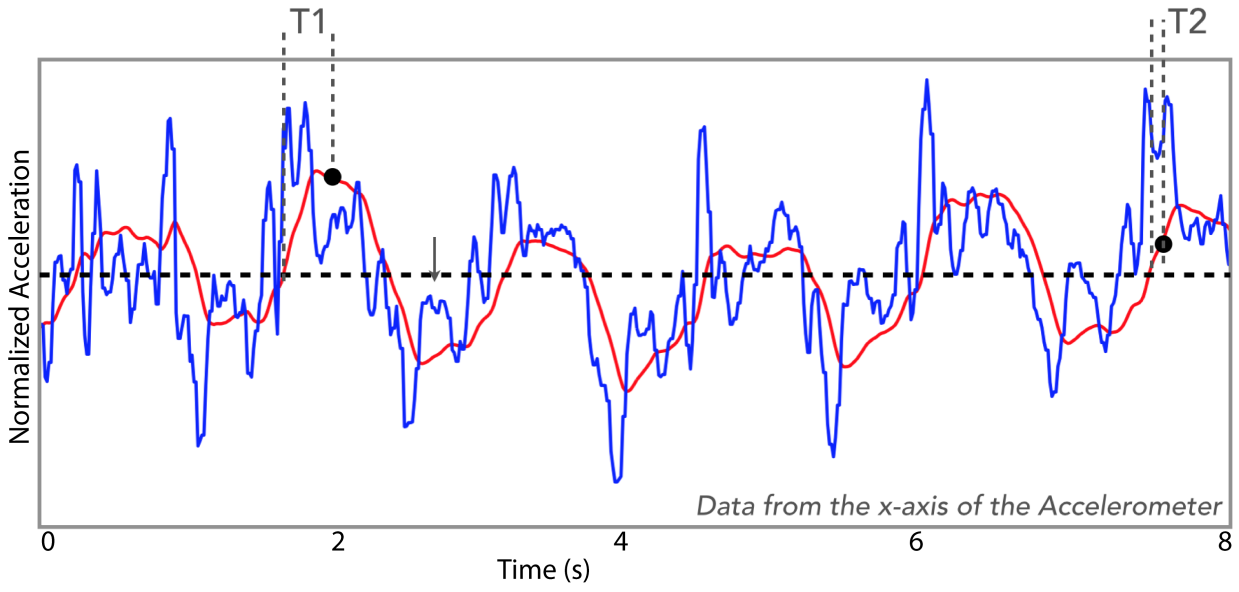

Figure 3.3: The blue waveform is the raw acceleration (100 Hz) of the phone along the *x*-axis when the user was walking. Red waveform is the low-pass filtered version of acceleration data. T1 and T2 are the elapsed times since the signal crossed the mean in the upward direction and the user tapped on the screen.

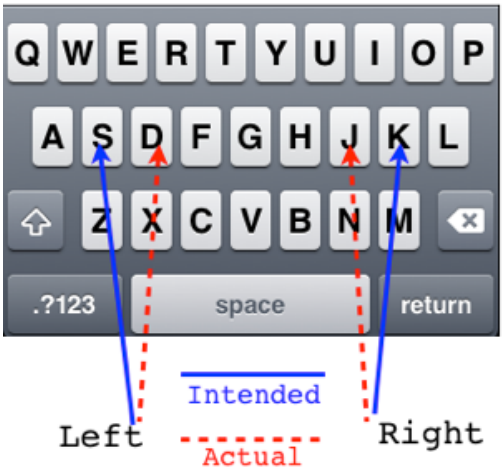

Figure 3.4: The Walking Pattern Model takes into account error in typing immediately after the user's left or right foot strikes the ground.

Other features for the classifier includes the tap location on the screen, the direction in which the phone is going in y- and z-axes, and the last foot that struck the ground. The classifier also expects three temporal components denoting time since the last change in direction in the three axes. On 10-fold cross-validation with the Collect data, the Walking Pattern Model outperformed the Displacement and Acceleration Model with a mean classification accuracy of 95.7% compared to 94.6%, a significant difference ($t_{15} = 5.11, p < .001$).

3.1.1.4 Combined WalkType Model

The Combined WalkType Model is a composite of the three sub-models: the Displacement and Acceleration Model, the Walking Pattern Model, and the Euclidean Model. A majority voting approach is used, whereby for each finger-touch, the key selected by at least two of the three internal models is output to the text stream. When all three models disagree, the Walking Pattern Model prevails, since it model performed the best in isolation on the WalkType Collect data. Figure 3.5 shows a block diagram detailing the approach. The Euclidian Model is included because, although classification accuracy was high for both of the accelerometer-based models, some keys become major sources of errors as they got masked by adjacent keys. An example confusion matrix is shown in Figure 3.6. Here a

more frequently occurring key dominates adjacent keys, for example, “A” dominates “S”. To counter this problem WalkType combines the two acceleration models with the Euclidean model. As mentioned earlier, the Euclidean model selects the key containing tap location. Although simple and non-adaptive, this model increases the probability of less-frequently occurring keys like “W” being correctly classified.

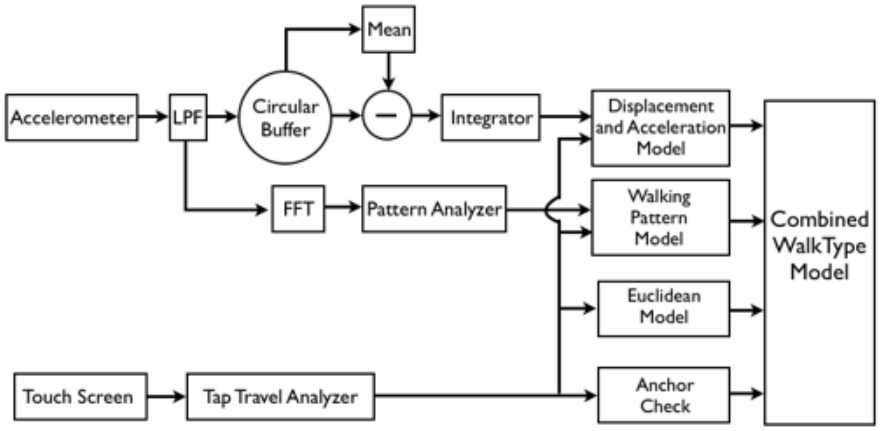

Figure 3.5: Block diagram of major components of WalkType’s Model Building phase.

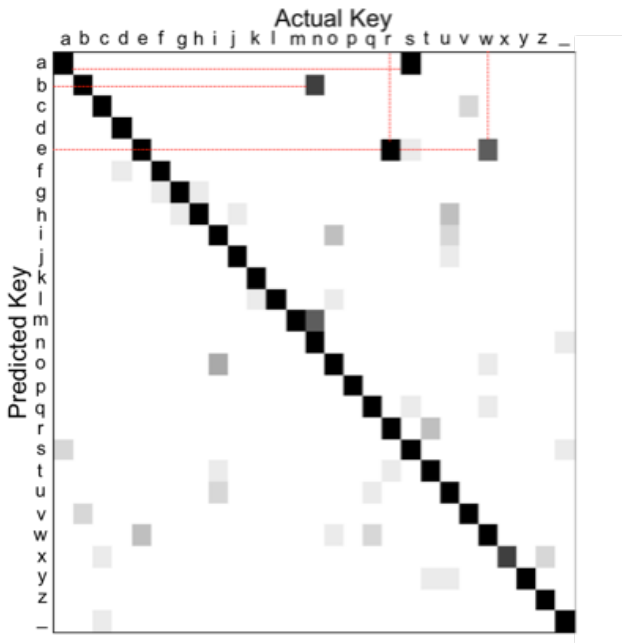

Figure 3.6: Frequently occurring keys dominate adjacent keys. ‘s’ is dominated by ‘a’, ‘r’ ‘w’ are dominated by ‘e’.

The mean classification accuracy of the Combined WalkType Model is 97%, significantly higher than both the Displacement and Acceleration Model and the Walking Pattern Model (paired two-tailed t-tests, respectively: $t_{15} = 6.51$, $p = .001$; $t_{15} = 5.94$, $p < .001$).

The Combined Walktype model also incorporates key-target anchoring [78]. After incorporating the three models, there were still taps that, although clearly landing in a key's center, did not get classified correctly. Gunawardana*et al.* [78] addressed this issue in the context of adjusting key-press probabilities based on a language model. Their work demonstrated that anchoring some part of the key increases overall typing accuracy. WalkType thus defines an anchor area in the middle of each visual key; a tap within that anchor area bypasses the classification models and instead returns the visual key.It reserves the central 20% along the x-axis and 50% along the y-axis of each key as the anchor area. Introducing anchors further increased the overall accuracy of the WalkType Combined model to 97.28% (significant compared to without anchors: $t_{15} = 5.193$, $p < .001$).

This final model was used in WalkType. Referring back to Figure 3.2 shows the cumulative improvement in accuracy for each component within this final model.

3.1.2 Evaluating the Realtime System

While the performance of the Combined WalkType model demonstrated the potential of WalkType to improve key-press classification accuracy, it still wasn't clear if the performance benefits would reflect in a real typing task. I implemented a real-time smartphone keyboard for Apple iPhone 3Gs that was informed by the Combined WalkType model. In a controlled evaluation, I sought to see whether WalkType would lessen the text entry performance degradation incurred by walking compared to sitting.

3.1.2.1 Participants and Apparatus

Sixteen participants (8 male, 8 female) ranging in age from 21 to 40 years ($M = 29.7$, $SD = 5.7$) were recruited. Five of these participants also participated in the

model-building study. All participants had more than 10 years of experience with computers and self-rated as intermediate to expert computer users. Fourteen participants were near-expert touch screen smartphone users with approximately 2-3 years' worth of use. Two participants did not own touch screen smartphones and had little experience with them.

Participants used the custom experiment software on an Apple iPhone 3GS that has a 3.5-inch capacitive screen with 480 × 380 pixels. Figure 3.7 shows a screenshot. Because WalkType uses finger-travel and elapsed time as features, the decision of which key was pressed occurs when the user lifts his or her finger. Accordingly, I modified the key-press feedback to occur when the finger is lifted, as opposed to when the user touches the screen, as occurs for the built-in iPhone keyboard.

3.1.2.2 Procedure

The procedure was designed to fit in a single 45-minute session. Each session began with an introduction to the tasks and experiment software. Participants were asked to

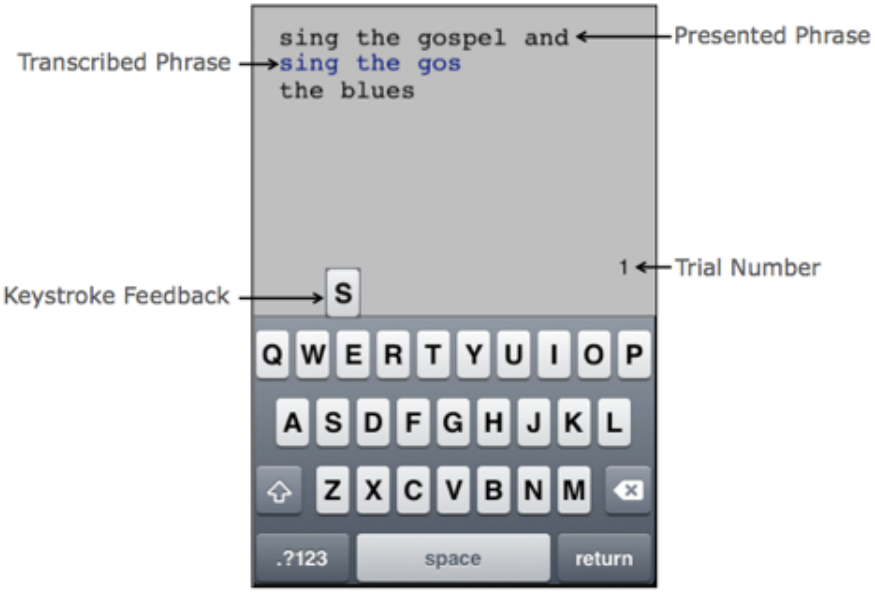

Figure 3.7: Testing interface showing the presented phrase, transcribed phrase, trial number, and keystroke feedback

familiarize themselves with the application and to ask any clarifying questions. This learning phase lasted approximately 5 minutes, until the user was comfortable with the system and had completed at least 5 phrases without assistance. For the walking tasks, we followed an approach similar to Kane *et al.* [108], wherein participants followed a human pacesetter, remaining within 3-5 feet. To simulate a routine and unconstrained environment, the pacesetter ensured that the walking speed was consistent at about 1.07 m/s, a pace that comfortably accommodates a wide range of age and abilities [68]. The walking tasks were performed in a relatively quiet corridor of a university building.

For each condition, participants completed 30 test phrases, 24 of which were randomly selected from the MacKenzie and Soukoreff phrase set [149]. Apart from these, every fifth phrase was a randomly selected pangram to cover all letters in the alphabet. Participants were asked to hold the phone in both hands and type with their thumbs. Participants were asked to type quickly and accurately, and to fix errors unless those errors were noticed "far behind" their current point of entry. Finally, participants were requested to complete each trial without failing to keep walking and were offered a rest period at the end of each trial.

3.1.2.3 Design and Analysis

The study was a within-subjects 2×2 factorial design. The factors and levels were:

- ***Interface:*** *WalkType* and *Control*. Both interfaces used the same experiment software but the underlying key-press classification models were different. WalkType used the final combined model from the previous section, while Control was non-adaptive, using only the Euclidian model.
- ***Posture:*** *Walking* and *Sitting*.

Presentation of the interfaces was counterbalanced. Within each interface, postures were also counterbalanced. With 30 trials (test phrases) in each condition, participants performed 2×2×30=120 trials each, for a total of 1920 trials in the study. Overall, we collected 57,663 key presses from 16 participants.

The main measures were speed, calculated as words per minute (WPM), and uncorrected error rate, following Soukoreff and MacKenzie [210]. Uncorrected errors represent those errors left in the transcribed text. Corrected errors, which are errors made during entry, are of less interest, as such errors slow WPM and are thus subsumed by it. Also, to evaluate whether participants could perceive any difference between the two systems, at the end of the session we asked them to rate which one of the two systems they preferred and why. To guard against any bias, participants were not initially made aware of which keyboard was Control and which was WalkType.

StreamAnalyzer [236] was used to calculate WPM and uncorrected error rate. The effects of presentation order was tested on the main measure of typing speed using a 3-way ANOVA with *presentation order* of the interfaces as a between-subjects factor and *Interface* and *Posture* as within-subjects factors. No main effect of presentation order was found, indicating that overall counterbalancing was effective. However, there was an asymmetric skill transfer: a significant interaction occurred between presentation order and Interface on typing speed ($F_{1,14} = 5.569$, $p = .032$, $\eta^2 = .288$). The control condition benefited more when it followed WalkType than vice versa.

For uncorrected error rate, I used the nonparametric Aligned Rank Transform [234] with *Interface* and *Posture* as within-subjects factors. We used this nonparametric procedure because uncorrected error rate is highly skewed toward zero and violates normality. All pairwise comparisons were protected against Type I error using a Bonferroni adjustment.

3.1.2.4 Performance

3.1.2.4.1 Speed (WPM).

Speed results are shown in Figure 3.8. Overall, WalkType improved typing speed regardless of whether the user was sitting or walking: on average 31.1 WPM ($SD = 10.7$) compared to 28.3 WPM ($SD = 9.6$) in the control condition. This difference was significant, as seen in a main effect of Interface on typing speed ($F_{1,15} = 6.777$, $p = .020$, $\eta^2 = .311$). No significant main effect of Posture was found.

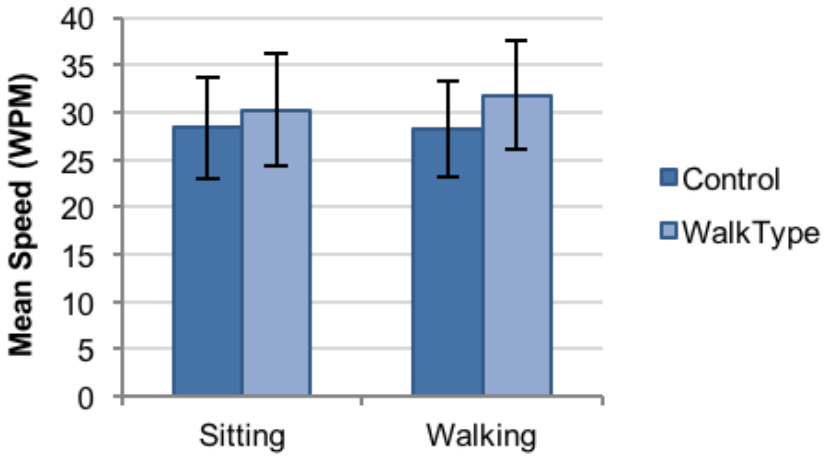

Figure 3.8: WalkType resulted in higher typing speeds than the control condition, particularly while participants were walking. Error bars are 95% confidence intervals.

WalkType was expected to improve performance more for walking than for sitting, which would be seen through an interaction of Interface × Posture on typing speed. This interaction was only a trend ($F_{1,15} = 3.485$, $p = .082$, $\eta^2 = .189$). Based on our hypotheses, however, I conducted pairwise comparisons of the two interfaces within each level of Posture. Pairwise comparisons showed that participants benefited more using WalkType while walking than while sitting. For walking, WalkType improved typing speed by 12.9% compared to the control interface, which was a significant difference ($p = .002$). For sitting, in contrast, no significant difference was found between the two interfaces ($p = .166$).

3.1.2.4.2 Uncorrected Error Rate.

Figure 3.9 shows mean error rates per condition. Mirroring the speed results, participants exhibited a marked improvement in error rate while using WalkType. A main effect was found for Interface on error rate, indicating that, overall, WalkType significantly reduced errors compared to the control condition ($F_{1,15} = 22.339$, $p < .001$, $\eta^2 = .598$). Posture also significantly affected error rate, with walking resulting in increased errors compared to sitting ($F_{1,15} = 9.316$, $p = .008$, $\eta^2 = .383$).

Of perhaps more interest is how WalkType impacted errors compared to the control

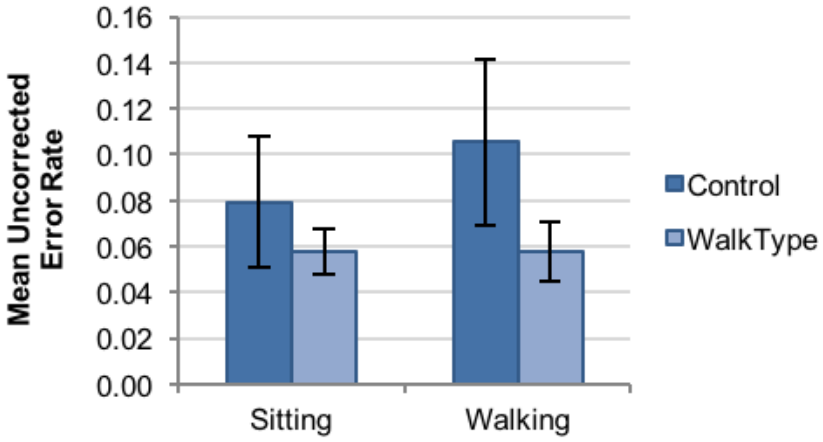

Figure 3.9: WalkType resulted in lower error rates than Control, especially for walking. Error bars are 95% confidence intervals.

condition while walking. A significant Interface × Posture interaction for error rate showed that the effectiveness of the interfaces differed by posture ($F_{1,15} = 13.139$, $p = .002$, $\eta^2 = .472$). Pairwise comparisons revealed that WalkType was particularly effective at accommodating the situational impairments introduced by movement. Compared to the control condition, error rates decreased from 10.5% to 5.8% with WalkType, which was a significant difference ($p = .004$). No significant difference in error rate was found between the two interfaces for sitting.

3.1.2.4.3 Preference.

At the end of the study I also asked the participants which of the two interfaces they preferred. Preferences reflected performance results, with 14 out of 16 participants choosing WalkType over Control ($\chi^2_{(1,N=16)} = 7.56$, $p = .006$).

3.1.3 Discussion

The goal here was to develop an adaptive soft keyboard that leverages accelerometer data to compensate for the situational impairments introduced while walking. WalkType successfully improved both typing speed and error rates, particularly for users walking. On average, WalkType improved typing speed by 12.9% and reduced uncorrected error

rates by 45.2% while participants were walking. Although there were no visual differences between the control interface and WalkType, participants perceived the performance benefit of WalkType and overwhelmingly preferred it to the control condition.

The Combined WalkType Model uses a majority voting approach between the three models. We also tried to combine these models into one single model with one decision tree, but the performance of the system decreased considerably because the number of features increased and became unwieldy.

The WalkType models were built using data collected both while participants were walking and while they were sitting. While experimenting with different models, I observed that using only the training data from walking further increased classification accuracy for walking, but decreased accuracy for sitting. Based on this finding, I combined the two datasets for the final WalkType model. However, detecting whether a user is walking or sitting and dynamically switching between different models—one trained on walking data and one trained on sitting data—would likely provide a further performance benefit.

The analysis showed that the major source of incorrect classification was confusion between adjacent keys in the same row. I created a confusion matrix on the basis of our classification results and found that 72.8% of incorrect classifications were of adjacent keys. Further examination into whether misclassifications occurred more towards the left or right sides of the keyboard showed that misclassifications occurred about evenly in this regard. The split of same-row errors was 48.4% to the right of the intended key and 51.6% to the left. This predominance of same-row confusion shows that it was easier for users to reliably hit in the vertical direction compared to the horizontal direction. That users could do this was somewhat unexpected. While walking, most vibrations are along the y-axis, *i.e.,* the phone moves backward and forward more relative to the user. I anticipated, incorrectly, that this would lead to more inter-row misclassifications. (Thankfully, WalkType is data-driven, and thus it did not suffer for our misconception.)

Most researchers examining effects of walking on user interfaces do not use a pacesetter,

but either ask participants to "walk normally" or employ a treadmill. Kane *et al.* [108] also used the pacesetter approach to make sure that participants walked at near constant speeds, while enabling them to be "off the treadmill" and in a natural context. Some of the features incorporated in WalkType, such as the dominant frequency of the user's walking pattern and its amplitude, may be useful in adapting to different walking speeds. Additional testing is needed to determine how well WalkType performs with varying walking speeds and whether additional training data is needed to model that context.

During the model-building phase, I collected typing data in a condition where users were given feedback only as to whether they had hit a key, but not whether they hit the correct key. In comparison, in the final user study, participants saw which letters they had entered, which could have led them to more accurate typing behavior. This leads to an interesting conjecture—that training the system on more realistic data, like I collected in our evaluation, could further improve performance.

Finally, the study here concentrates on the scenario where users hold the phone in both hands and type with their thumbs. This brings us to an interesting juncture. Research has shown that user's typing behavior changes with their hand posture as well [13]. Therefore, it will also be interesting to explore how the models perform on data collected with users holding the phone in one hand and typing with the index finger of the other hand, and so on. In the next section I will discuss the effect of hand posture on typing accuracy and how can the sensors on the mobile devices help.

3.2 Hand Posture

While motion can be a significant contextual factor that affects mobile device use and cause situational impairments [200], another very significant factor is the user's hand posture; *i.e.,* the grip with which the user manipulates a mobile device. Research has shown that hand postures including grip, hand pose, the number of fingers used, and so on significantly affect performance and usage of mobile devices [13,237]. For example, research has shown that the index finger is much better at selecting targets on a screen when compared to

the performance of the user's thumb. Similarly the two-handed posture is also better in pointing performance versus one hand. It can also be argued that for most users their dominant hand will be better than the non-dominant one. So overall the performance a user achieves while interacting with a mobile device can be greatly affected by their hand posture. And yet, our devices have almost no clue how they are being held or manipulated, and therefore cannot adapt and respond appropriately with an adaptable user interface.

This has been an area of keen interest for the HCI researchers and various technique have been explored to accommodate some of these interaction challenges, like the change in device orientation due to hand movement [39, 94]. Most of the past approaches investigated the possibility of adding some new hardware to the device to make it adapt to these changes. However, this dissertation focuses on extending the capabilities of the sensors that are already their on the mobile device. Therefore, in this section we will discuss an approach that uses a combination of the device's touchscreen and the built-in inertial sensors (gyroscope and accelerometer), and the vibration motor to infer the user's hand posture and adapt the interface accordingly. This section is divided into two parts: The first part discusses how we can use the on-device sensors to infer the user's hand grip (section 3.2.1) and the second section goes into the details of adapting to the user's changing hand postures (section 3.2.2).

3.2.1 Inferring Hand Posture

In this section I discuss *GripSense*, a system that uses the touchscreen, accelerometer, and gyroscope to infer the user's hand posture. GripSense detects hand postures over the course of a small number of interaction steps (*e.g.,* , tapping, swiping the screen). It infers postures such as the use of a finger, left thumb, right thumb, and which hand is holding the device. GripSense performs its sensing by measuring the device's rotation, tap sizes, and the arc of swiping motions. Previous work on hand-posture detection has leveraged accelerometers for detecting whether a device is used in a stationary environment, in a hand, on a table, or in motion [198], and researchers have also used *external sensors* for grip detection [84,

116, 214]. GripSense is the first work that explored the use of just the on-device sensors to detect the user's hand grip.

GripSense was evaluated in a controlled study with 10 participants. The evaluation showed that GripSense can differentiate between device usage in hand or on a flat surface with 99.7% accuracy and various hand postures with 94.3% accuracy and, on an average, makes a decision within 5 "interaction steps", *i.e.,* actions taken by the user that give GripSense information.

3.2.1.1 Previous Work

Prior to GripSense, others have also proposed techniques for detecting hand postures. Kim *et al.* [116] and Harrison *et al.* [84] used capacitive touch sensors to differentiate between numerous grips. Taylor and Bove [214] additionally leveraged accelerometers to dynamically detect changes in a user's grip for improved interactions. GripSense, in contrast, requires no additional instrumentation of a modern smartphones to robustly detect hand grips. The main trade-off for this capability is that the user needs to be interacting with the device for GripSense to make its inferences.

Holz *et al.* [99] evaluated systematic error in target selection due to change in finger posture. A number of researchers [94, 113, 230] have suggested that although the user usually prefers single-handed operation while using smartphones, traditional mobile interfaces are designed for two-handed operation. Karlson *et al.* [112] studies such interfaces and evaluated how they impede thumb-based usage. Azenkot and Zhai [13] found that different hand postures induced different touch patterns and affected overall performance while typing on a mobile touchscreen keyboard. In the Section 3.2.2, when talking about adapting to the changes in hand posture, we will discuss this phenomenon in more detail.

Table 3.1: Summary of all inferences made by GripSense, and when and what features were used for each of the inferences.

Inference	Features Used	Sensor Event	Latency
Table vs. Hand	Gyroscope (Low frequency in all axes)	Touch Down	1
Thumb vs. Index Finger	Gyroscope (Low frequency in x- and y axis	Touch Down	3
	Swipe Shape	Touch Up	
	Touch Size	Touch Down	
Left Thumb vs. Right Thumb	GyroScope (Low Frequency in y-axis)	Touch Down	5
	Swipe Shape	Touch Up	
	Touch Size	Touch Down	

3.2.1.2 Approach

GripSense uses multiple sources of information to detect a user's hand posture. Among these sources is the data from the device's built-in gyroscope and touchscreen. GripSense infers where the phone is *(a)* in a user's left hand and operated with left thumb, *(b)* in a user's right hand and operated with the right thumb, *(c)* in either hand and operated with the index finger of the other hand, *(d)* on a flat surface, or *(e)* being only grasped by the user and not operated. It uses a combination of three features: (1) relative variance in rotation, 3) change in touch size, and (3) direction of arc for finger swipes. In this section, I outline the concept and theory behind GripSense.

3.2.1.2.1 Rotation of the Device

The first feature is the rotational movement of the device as the user touches the screen. In a one-handed interaction, the phone rotates in response to touches at the top of the screen more than it does to touches at the bottom of the screen (Figure 3.10). This is to compensate for the limited range of the thumb; fingers move the device as the thumb extends to reach the top of the screen. In contrast, touches at the bottom of the screen result in less angular motion because that area is usually within the thumb's range. When the user interacts using

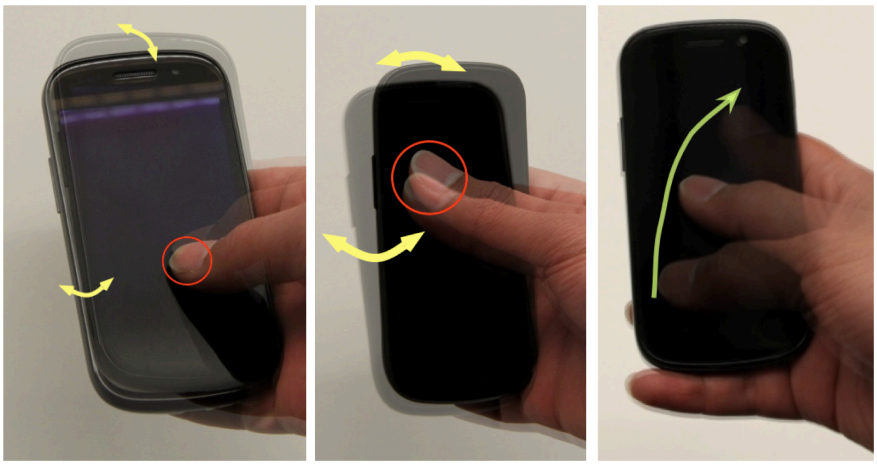

Figure 3.10: ***(left)*** **Minimal device rotation in x- and y-axis, and smaller touch size when the user touches nearby the thumb.** ***(center)*** **Significantly more rotation in x- and y-axis and larger touch size when the far quadrant of the screen is touched.** ***(right)*** **The shape of the swipe arc in the case of right thumb. (All of these phenomena are mirror-images for the left thumb.)**

their index finger, there is no difference in the angular motion from the touches at the top or bottom of the screen. If the device is on a table then there is no change in any of these parameters before the touch event is registered.

To leverage these insights, we store the angular velocities around the x-axis sampled at 1 kHz from the gyroscope in a 250 ms buffer. The data in the buffer is then passed through a low-pass filter to isolate the low frequency angular velocities. GripSense records the last two angular velocities observed for touches in the top third of the screen and the bottom third of the screen (determined from a pilot with four users). If the difference in variance of angular velocities for touches in the top is five times greater than for touches in the bottom of the screen, GripSense assumes that it was thumb-based interaction. If the difference in the variances does not exceed the threshold for three consecutive touches, then the system baises the final decision towards selecting the "index finger".

Similarly when a user holds the phone in their left hand and interacts with their thumb, the touches on the right side of the screen cause significantly more angular motion than the touches that are nearer to the palm; again because of the copensation for the limited reach of the thumb (Figure 3.10). The phenomena is mirrored for the right hand. In case where

Figure 3.11: (left) Touch size is bigger when a far side is touched. (right) Touch size is smaller when a near side is touched.

the system infers a thumb-based grip, it uses a similar approach as before, except now it logs the variance for the *y*-axis of the gyroscope for touches on the left third of the screen and the right third of the screen. If the variance in angular velocity of the last two touches on the left side is greater than that on the right side, then it assumes that the phone is in the user's right hand. Moreoever, if the difference in angular velocities is more than ten times greater in consecutive touches, it sets a "high confidence flag". The importance of this flag will be discussed later.

3.2.1.2.2 Touch Size

The second feature for grip detection is based on the change of size of touch recorded on the touch screen. When a user interacts with the sides of a screen, the shape of the thumb and the rotation of the device in the user's hand leads to changes in touch size and shape. The touch size on the same side as the thumb will be smaller than the touch size of the far side from the thumb (Figure 3.11).

For this feature, the screen is divided into six (2x3) parts and keep track of the last two touch sizes. Note that the Android platform provides a method to get the touch size on the screen, apart from the *x*-*y* location of touch. GripSense compares the touch sizes in the left and right third of the screen for the same third of the screen height. If the difference in the mean of the touch sizes is more than 25%, the system is biased towards a thumb-based interaction. The rationale here is that the shape of the thumb leads to a very different

touch size when the user interacts with the far side of the screen as compared to the one on the near side. When the user extends their thumb to interact with a farther portion, they touch the screen with their pad of the thumb, whereas touches closer to the screen are usually performed with the thumb tip. If the larger tap size is on the left side, then GripSense infers a right hand grip, and vice versa. Moreover, if the differences in touch sizes is more than 40% for consecutive touches, the heuristic sets a "high confidence flag". If the difference is less than 25%, it biases toward index finger-based interaction.

3.2.1.2.3 Shape of the Swipe Arc

This feature is only applicable when the user swipes on the screen. Because of the shape and position of the thumb, users often draw an exaggerated arc instead of a relatively straight line. Karlson *et al.* [113] observed similar arcs and analyzed how an interface can be more effective by biasing all the interactions along/within this arc. GripSense uses this arc as its "signal" to detect the user's hand posture. The initial pilot study showed while using the phone with the index finger there is no consistent arc. However with the thumb, there is a consistent and exaggerated arc to the right or left depending on which thumb is being used. Figure 3.10 shows the arc formed by the right thumb while performing a bottom-to-top swipe. A mirror image of this arc will form in the case of the left thumb.

If the differences in coordinated of the start and end position of a vertical swipe are more than 5% of the screen resolution, GripSense biases itself towards one of the two thumb postures. Even so, I observed that sometimes a thumb-based swipe does not result in an arc. Instead, the phone experiences angular motion in the hand. For example, a right-handed swipe from bottom to top results in a counter-clockwise rotation. These two phenomena combine to form a robust heuristic for handing posture detect in case of swipe. As with the other two heuristics (phone rotation and touch size), the final within-heuristic decision is made when the system biases toward the same posture twice in a row.

3.2.1.2.4 Making the Final Decision

If swipes are present, GripSense uses majority voting on the output of each heuistic to

decide the posture. If all three votes diagree, the posture is marked as "unknown". In the absence of swipe, a final decision is made only if both touch size and rotation heuristics agree and if the "high confidence flag" in one of the heuristics is set. If both heuristics come up with different decisions, then the system chooses the heuristic with a "high confidence flag". If none of the heuristics is confident, then the posture is set to "unknown" and the decision is deferred until the next interaction step.

3.2.1.2.5 Two-Thumbed Hand Posture Detection

Tap sizes increase as the user touches the far side of the screen, *i.e.,* when operated with the left thumb, the areas touched on the right side of the screen will be bigger than those on the left side and vice versa. In the pilot study, I observed that in case of two-thumbed posture, the tap sizes in the center of the screen are at least 25% larger than those on either side. Another heuristic that is combined with tap sizes is the relative difference in time elapsed between taps on either side of the screen. When using one hand to operate the phone, it takes longer to go from one side of the screen to the other. Hence, if a tap on one side of the screen is followed by another tap on the opposite side, the time difference will be larger than the average time difference because the thumb/finger needs to travel farther. In contrast, when operated with two thumbs, the time difference between taps on opposite sides of the screen will be significantly less. Hence, the system inferred two-thumbed interaction if the difference in time interval between taps on opposite sides and the mean time interval between taps was greater than 30%.

3.2.1.3 Evaluation

I evaluated the performance of GripSense with ten participants; ranging in age from 21 to 32 years M=26.9, SD=3.6). A Samsung Nexus S phone was used for the evaluation and the device's angular velocity was recorded at 1kHz using the built-in gyroscope. Considering GripSense has separate approaches for tap and swipe events, the participants were asked to perform both gestures in different applications. One application, *Contact Selection App*, was more scrolling heavy than tapping; the other, *Text Entry App*, used

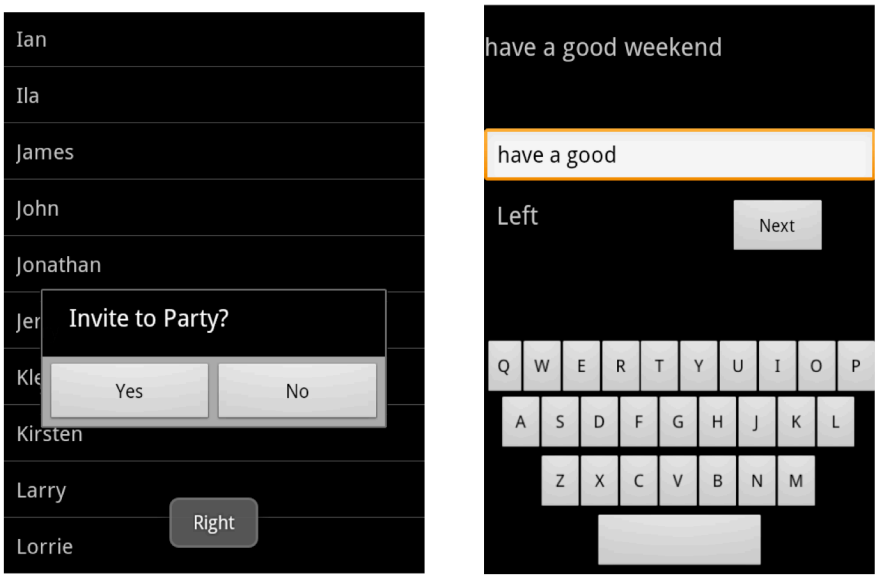

Figure 3.12: (left) Contact Selection App. Swipe-intensive application that helps in quantifying swipe heuristic performance. The pop-up in the bottom center of the screen prompts the user with the current intended hand posture. (right) Text Entry App. Tap-intensive app helps in evaluating performance in absence of swipes. The left area of the screen, just below the text field prompts the user with the current hand posture.

only taps (Figure 3.12. The data collection sessions lasted approximately 30 minutes. The Contact Selection App presented the participants with a list of 100 random names. The investigator asked each participant to select 50 of these names in a random order. Then, in order to collect data for taps on both left and right side of the screen, the participants were asked to randomly click on "Yes" or "No" in the next dialog box (Figure 3.12). After every 5 name selections, the app prompted the participants to switch postures. The order of the postures was randomly generated. The Text Entry App required only tapping as it was basically a custom keyboard. Participants were asked to enter 15 short English language phrases randomly selected from the MacKenzie and Soukoreff's phrase set [149]. At the end of each phrase, the application instructed participants to switch to a new randomly selected posture.

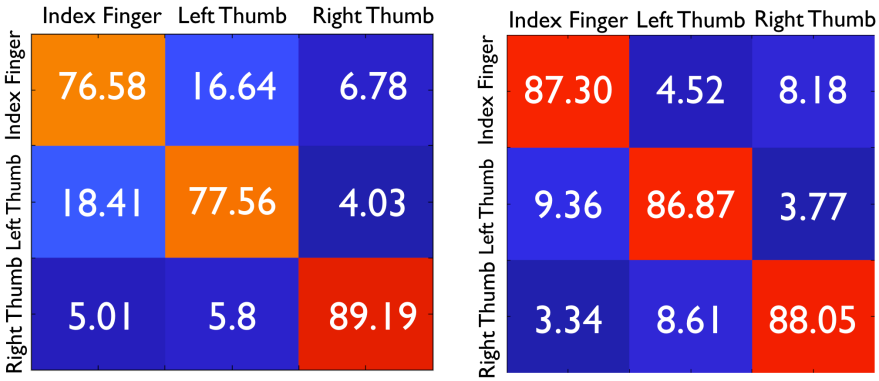

Figure 3.13: (left) Confusion matrix for classification of hand posture using the Text Entry App. The *x*-axis shows the classification and the *y*-axis shows the actual posture. (right) Confusion matrix while using the Contact Selection App.

3.2.1.4 Results

Figure 3.13 shows the results from the controlled user study. It shows the confusion matrices for the three hand postures. The *left* figure shows the performance while the users were using the Text Entry App (81.11% classification accuracy), and the *right* figure shows the performance of the Contact selection app (87.4% accuracy). While using the text entry app, GripSense made its decision after, on average, approximately 5 interaction steps. The number of interaction steps required to make the decision could be decreased but that would be at the cost of accuracy. The trade-off can be tuned depending on the actual application that uses it. The performance, expectedly, improved further while using the Contact Selection app, as it permits the use of the third heuristic, the shape of the swipe arc. The accuracy improved to 87.4%. In this case, GripSense was able to make a decision on hand posture after about 4 interaction steps.

3.2.1.5 Discussion

GripSense successfully infers a user's hand postures with high accuracy. But, there is slight degradation in the performance of GripSense when deciding between left-handed thumb-based operation and index finger-based operation. I believe this degradation is due to the system confusing one-handed and two-handed operation. This problem is

mitigated to a large extent in the swipe-heavy application, Contact Selection App. In the case of the keyboard application, the user's interaction was largely limited to the bottom half of the screen and GripSense depends on analyzing differences in device movement when interacting with different parts of the screen. Most probably this degradation would not be felt in real world applications that require users to interact with various parts of screen and hence provide much richer data for the algorithms. However, the keyboard app provides ample interaction switching between the left and right sides of the screen, so GripSense's performance is not dramatically affected.

Extended use of the touchscreen, gyroscope, and vibration motor can have significant power implications. But I only sample these sensors when the device is interacted with. I did not do a direct analysis of energy use, but anecdotally there was no significant reduction in battery life during the user studies. If needed, future iterations of these systems could employ a multistage approach by sampling at a low frequency first and then higher rates, as needed. That said, modern devices use a motion co-processor to sample these sensors and it promises to drastically reduce the power drain due to constant sensor sampling.

3.2.2 Adapting to Hand Posture

GripSense provides us with a way to efficiently and seamlessly detect the user's hand posture. By simply using the sensors that are already on the device, the system does not impose any initial burden on the user. One of the primary messages of this dissertation is that the on-device sensors can be used to seamlessly improve the user experience. However, it still isn't clear how the user experience can be improved using this new information provided by GripSense. [94] suggested to use the sensors to detect the device's orientation and orient the phone's display. This feature is now standard across all mobile devices and it is impossible to find a mobile device that does not reorient itself using the information from the device's inertial sensors. I believe that the hand posture information can also act "behind the scene" and seamlessly improve the way we interact with our devices. AppLens and LaunchTiles [113] designed new interfaces mindful of the

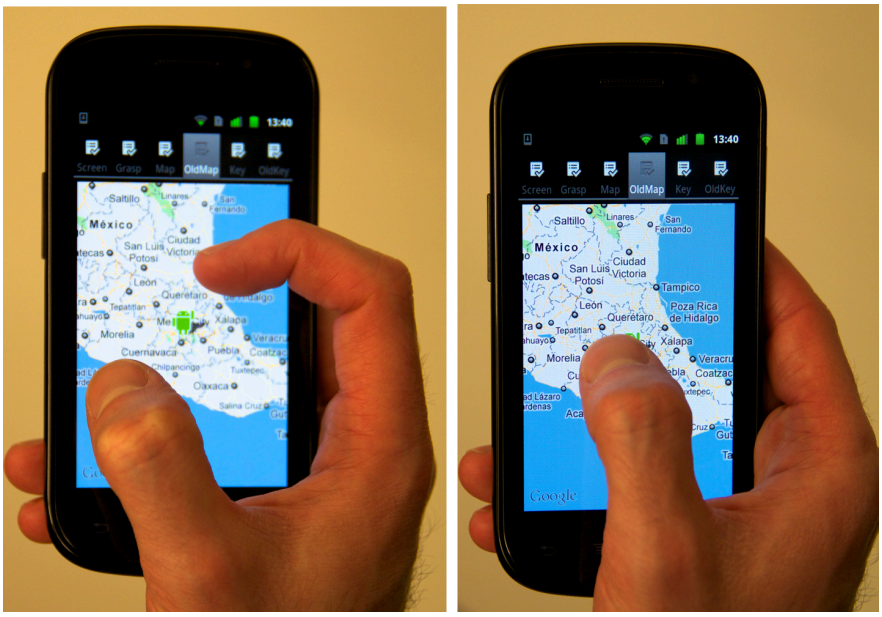

Figure 3.14: (left) It is difficult for a user to perform interactions like *pinch-to-zoom* with one hand. (right) UnderPressure gets the grip information from GripSense and infers the pressure exerted on the screen to facilitate new interactions like zoom-in and zoom-out.

limited precision and range-of-motion of the user's thumb. Additionally, Azenkot and Zhai [13] found that different hand postures induced different touch patterns and affected the overall performance while typing on a mobile touchscreen. In the rest of this chapter, I will discuss two approaches that build on top of the earlier work done in this area. The first approach, called *UnderPressure*, uses GripSense and whenever the user is using the phone with one hand, it enables a new way to interact with the device: *pressure input*. The second approach, called *ContextType*, uses GripSense and adjusts the keyboard's touch map (not visible to the user) accordingly. ContextType reduces typing errors on the phone by approximately 20% by simply adding the 2-bit information about the user's hand posture.

3.2.2.1 Pressure-based Input

UnderPressure leverages the built-in vibration motors in a new way to help infer the amount of pressure being applied to the screen when the user is interacting with the

phone. This information can be used to enable alternate interaction techniques with mobile devices. Some of the latest smartphones, *e.g.,* , Apple iPhone 6S, have a pressure-sensitive touchscreen to enable a similar functionality. UnderPressure brings the same functionality to rest of the devices. This thesis also discusses some of the applications that can be built on top of such capabilities and how the device can actually use it to augment the interaction when the user is in various situational impairments. As an example, UnderPressure allows users to zoom-in and zoom-out of maps using the pressure input. Imagine if the user is holding their phone with one hand and the other hand is busy (*e.g.,* , with a shopping bag). The phone can use GripSense to detect the one-handed posture, switch on the pressure input using UnderPressure, and now the user does not need to use an awkward grip to zoom-in and out of a map (Figure 3.14). There has been long and consistent interest in augmenting mobile devices with pressure input. Iwasaki *et al.* [103] measured typing pressure on laptop keyboards using on-device accelerometers. Pressure Widgets [182] used pressure-sensitive stylus for adding pressure-based interactions to PDAs. Clarkson *et al.* [42] instrumented a flip-phone with force sensitive resistors to infer continuous pressure applied by the user. Force Gestures [90] also added detection of tangential forces for richer user interaction. Unlike UnderPressure, most of these efforts required additional device instrumentation.

3.2.2.1.1 Previous Work

As with UnderPressure, there is significant previous work that does not require custom instrumentation. Hinckley*et al.* [94], and Heo and Lee [91] used the smartphone accelerometers to leverage touch-induced vibrations as a proxy for pressure. In contrast with UnderPressure, these approaches do not get a continuous measure of pressure applied by the user. They provide a coarse proxy of the pressure and only infer the initial velocity with which the user's finger strikes the screen. Although useful in a number of situations, the granularity and frequency of such inferences is limited. Additionally, Heo and Lee [91] also found that the requirement of increased speed-of-contract results in higher target selection error. This limitation does not impede UnderPressure.

Fitzmaurice *et al.* [67] coined the term "graspable user interfaces". Such interfaces allow devices to become more context-sensitive and thereby improve "expressiveness or the communication capacity" of the computer. Wimmer *et al.* [232] leveraged computer vision and optical fibers to detect grasping pressure on mobile device surfaces. Harrison *et al.* [84] leveraged FSRs to detect squeezing pressure. These works required additional device instrumentation to infer grip pressure. UnderPressure requires no additional hardware for its graspable user interfaces and is a completely software-enabled solution. Another software-only solution, one by Strachan and Murray-Smith [162] used muscle tremor as a proxy of pressure sensing in a squeezable interface. UnderPressure uses similar phenomenon and combines it with motor-induced vibrations to achieve more fine-grained estimation of pressure.

3.2.2.1.2 Approach

UnderPressure uses the gyroscope and vibration motor to classify the user's touchscreen touches into three pressure categories: *Light, Medium,* and *Heavy*. I observed that if we trigger the built-in vibration motor when a user touches the screen (similar to what is already done in a number of phones to provide haptic feedback), the user's hand absorbs a portion of these vibrations. This vibration absorption is proportional to the amount of pressure being applied on the screen (Figure 3.15). This damping effect is measured using the on-device gyroscope. I primarily look for the damping of vibrations induced by the vibration motor. However, I also observed that as the amount of force exerted increases, there is a subtle oscillating motion between the user's thumb and the four fingers that rest on the back of the device (see the low pass signal in Figure 3.15). I hypothesize that this oscillation occurs because the user's thumb and fingers try to continually compensate for pressure exerted and this oscillation has a much lower frequency compared to that induced by the vibration motor. UnderPressure uses both of these features to make a robust classification of the applied pressure. An effective combination of these touch-induced vibrations with damped motor-induced vibrations give a much more fine-grained and continuous proxy of pressure exerted on the screen. The custom

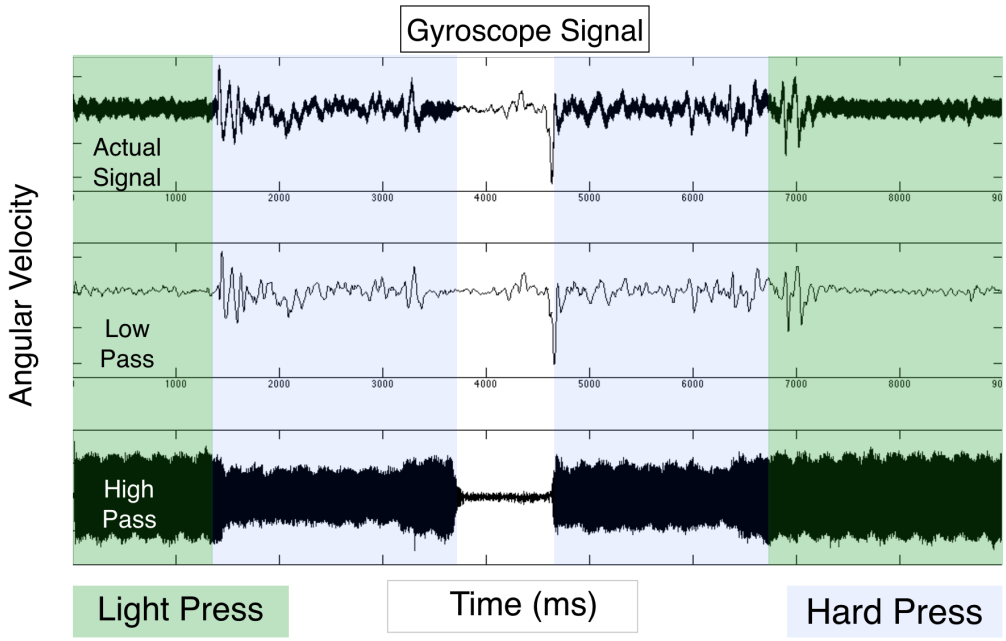

Figure 3.15: (top) Gyroscope signal when the user presses light, then hard, then waits for a second and presses hard and soft again. (middle) The lower frequencies generated from touch-induced vibrations increase with increase in pressure.(bottom) Motor-induced vibrations are diminished as the amount of pressure exerted increases.

Android app for UnderPressure gathered the angular velocities from the device's gyroscope at 1 kHz (Figure 3.15, top). Touch-induced vibrations can be obtained by passing this signal through a low-pass filter (Figure 3.15, middle), and the motor-induced vibrations through the high pass filter (Figure 3.15,bottom). In case of a hard press there is an exaggerated damping effect due to the vibrations getting absorbed by the user's hand. I calculate 90th percentile of the high frequency component of the observed signal and use it as an estimate for the damping effect. For the low frequency signal, I quantify the movement of the phone using the signal variance.

Essl *et al.* [63] have earlier used the size of touch on the screen as a proxy for pressure exerted, and UnderPressure also use this as a feature. An analysis of performance showed that this feature alone is a poor measure of pressure (only about 60% accurate in the pilot study), but when combined with vibration analysis it can marginally improve performance. The pilot study also showed that the amount of motor-induced vibrations absorbed by the

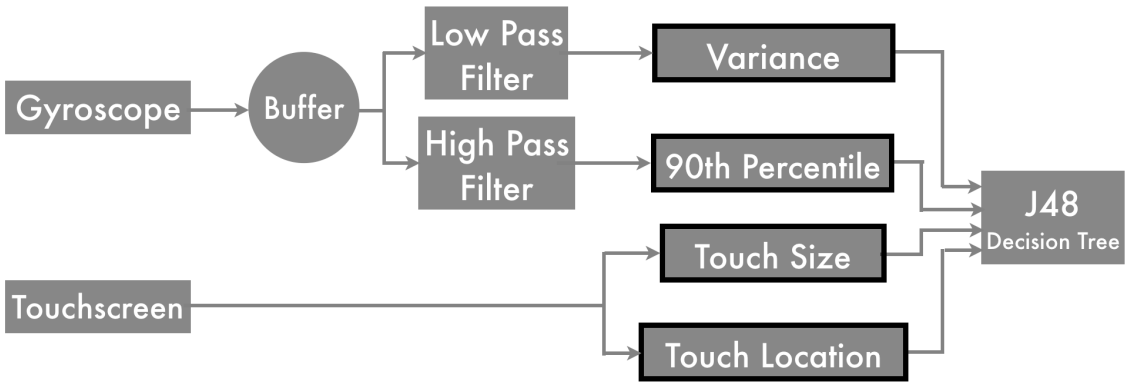

Figure 3.16: Block diagram of the major components of UnderPressure. Low frequency variance, 90th percentile of higher frequencies, touch size, and location are the features used for pressure-level classification.

hand was also dependent on the location of touch on the screen. Hence, UnderPressure divides the screen into a 4X6 matrix in portrait mode and added "touch zone" as another feature for pressure level classification. UnderPressure buffers the gyroscope daya at 1 kHz in a 500 ms buffer and analyze it every 250 ms (Figure 3.16). The data then passes through low pass and high pass filters and appropriate variances and 90th percentiles are calculated. These features, along with the touchscreen features (*zone* and *size*), were used to classify to one of three pressure levels using the Weka machine learning toolkit. Weka was used to generate J48 Decision Trees with pruning confidence set to Weka's default (0.25).

Using similar techniques as for quantifying pressure exerted on a touchscreen, UnderPressure also detects squeeze or grasp gestures. For example, imagine silencing a phone while it is still in a pocket or in a purse by squeezing it and without the need for fully retrieving the phone. Although grasping provides a significant amount of damping to the motor-induced vibrations, there was no significant variance in the low-frequency component of the gyroscope data; therefore, only higher frequencies were analyzed and their 90th percentiles were used as features for Weka's J48 decision trees.

3.2.2.1.3 Evaluation

The same 10 participants that were used for GripSense's evaluation were used to evaluate

UnderPressure as well. Participants used the custom Android app on a Samsung Nexus S. The device's angular velocities were recorded at 1 kHz using the built-in gyroscope. The groundtruth pressure values were obtained using thin-film force-sensitive resistors (FSR), affixed to the touchscreen. The decision tree model for pressure detection was learned and evaluated on the data collected from the participants in a 45-minute study each. Participants were asked to tap the screen in different hand postures detectable by GripSense: (1) thumb-based operation, (2) index finger operation, (3) on a table, (4) only held by the user but not operated. Twenty taps for each of three pressure-levels using each posture were recorded for all participants. Apart from the taps, the participants were also asked to perform seven longer, continuous touches lasting 10 seconds each for each pressure level. These longer touches were recorded because the pilot data showed that the low frequency touch-induced vibrations lasted for nearly 3 seconds and started attenuating thereafter as the hand became used to the pressure level.

The three different pressure levels were explained to the participants and they were asked to practice them. The groundtruth from the FSRs was visualized on the screen for feedback during training. Once the participants were comfortable entering three distinct levels of pressure, they were introduced to the data collection interface and procedure. Although all participants were comfortable distinguishing for themselves three pressure levels, absolute pressure values were not uniform across participants. The order of postures of postures in which data was collected was randomized to prevent unwanted effects from fatigue.

The procedure for collecting data to evaluate grasping pressure was same as that for collecting touchscreen pressure. Participants were asked to grip the device with three varying intensities. The least pressure was slightly less than what participants would apply while using their phone in general. The middle grip was meant to approximate how participants would normally hold their phones. In the highest pressure level, participants were asked to grip their phones tightly.

The participants were asked to press a button to trigger the vibration motor. The motor went off 5 seconds after the button was pressured. This gave the user ample time to get the

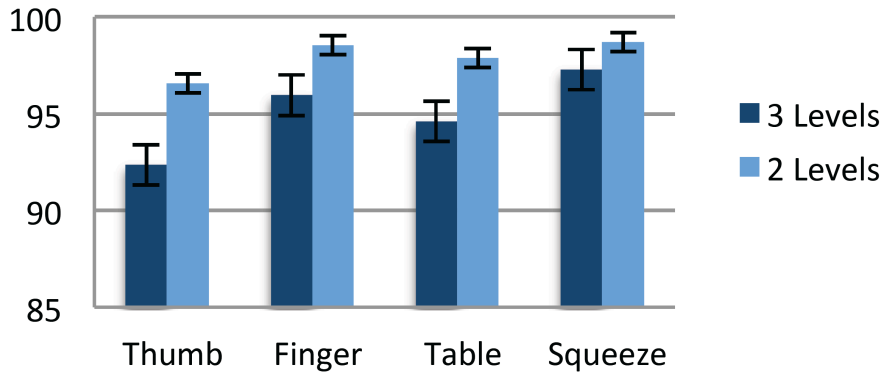

Figure 3.17: Squeeze gestures performed the best. Reducing the number of pressure levels to two improved the accuracy further. Error-bars are standard error.

phone in the correct grip. Then the device vibrated for 10 seconds. Five such tasks were run for each pressure level and for each participant.

3.2.2.1.4 Results

Figure 3.17 shows the accuracy of UnderPressure in detecting the pressure exerted on the phone in different postures. The overall accuracy across all postures was 95.05%. It may be that for many applications, two levels of pressure are adequate so I evaluated performance for two pressure levels as well. As expected, the performance of UnderPressure improved significantly to 97.91%.

As you may recall, I collected data for both taps (momentary touch) and long 10 seconds touches. The long touches facilitate functions like zooming-in and out of pictures, maps, *etc.* . During the pilot study I realized that different users have different ways of holding the phone and have different comprehensions of different pressure levels. Hence, I developed personalized pressure classification models for each participant. This meant that participants had to take part in a relatively longer data collection study. So I investigated how much data was enough to train the system for acceptable levels of accuracy, as one would want to have the smallest possible calibration sequence. Figure 3.18 shows the progression of improvement in average accuracy of UnderPressure as the amount of training data is increased. The graph has been divided into different

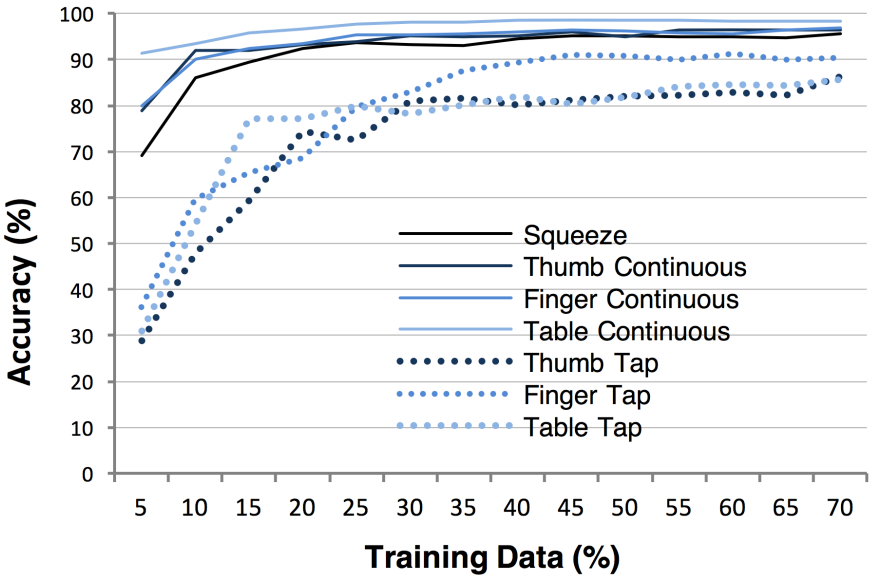

Figure 3.18: Improvement in average accuracy of pressure level classification for different gestures and touch-types with size of training data. Continuous touches reached maximum performance with significantly less training data.

accuracies for different gestures in variety of postures. The *x*-axis shows the percentage split in training and test data. Larger values on the *x*-axis signify more training data. It is clear from the figure that in the case of long, continuous touches, UnderPressure does not require a lot of training data. Even for momentary taps, in total, only 20 taps were recorded for each pressure level. In most cases, the classification accuracy reaches above 80% by the 30% split mark; meaning that only six taps for each pressure level were required to train a decision tree for each user.

Although this evaluation helped in understanding the feasibility of using a combination of GripSense and UnderPressure in sensing the user's hand posture and providing a new way of interacting with their touchscreens, that was not the end goal. GripSense and UnderPressure were developed with the goal to sense and accommodate for the situational impairments that the users experience when they use their mobile devices. In order to understand the effect of these technologies in circumstances where it is difficult to interact with the phone, I developed a map-navigation application in which the user could zoom in by pressing harder on the screen. The application first sensed how the user

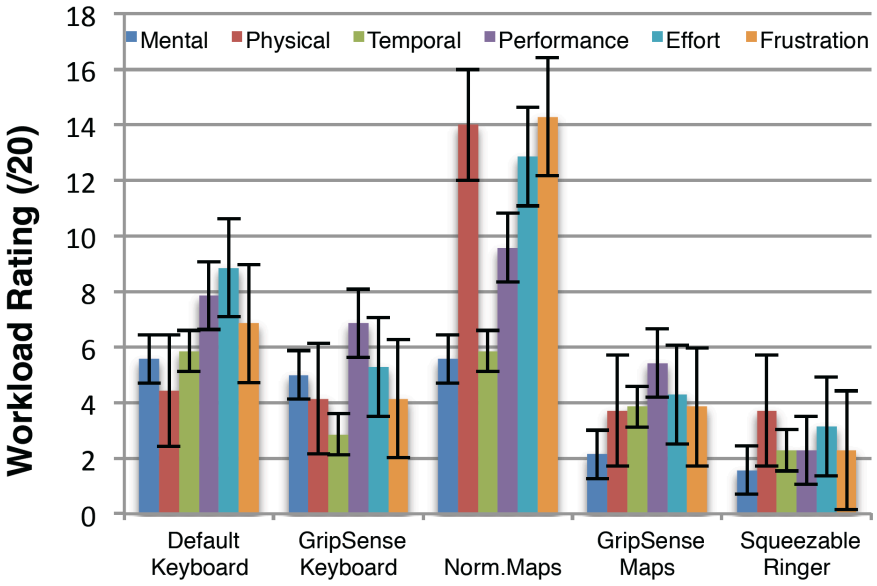

Figure 3.19: Perceived workload ratings show UnderPressure-based applications results in relatively low workload. Lower ratings are better. Error bars are standard error.

was holding the phone and if the app perceived that the user might only have one free hand, it switched to the pressure-based input for zooming-in and out of the maps (Figure 3.14). Hinckley *et al.* [97] have developed a similar application where they used hard and soft onset of taps for similar interactions. As noted previously, UnderPressure's approach differs because it does *continuous* pressure monitoring, not just detection of pressure at touch-onset, and therefore, the user does not have to impact the screen with increased velocity. In order to assess the usability of this Maps application, I presented the participants with a fully zoomed-out view of a world map and asked them to zoom-in to a set location. The participants needed to zoom in until they could see street names, and then they had to zoom out until they could see the entire county again. To mimic a situation impairment, I required participants to hold a coffee mug in one hand, and operate the device in their other hand. I also asked the participants to perform same tasks on an app that did not pressure-based input capabilities, and they had to use *pinch-and-zoom* to interact with the map using only one hand. After completion of a task for each system, the participants rated their perceived workload on a NASA TLS perceived workload index [88]. All 10 participants preferred using UnderPressure's

pressure-sensitive navigation, as it was much easier to navigate the app with one hand using pressure input to zoom in and out than to use one hand with pinch-to-zoom. On average, UnderPressure-based app did better than traditional app on all counts on the Likert scale (Figure 3.19). A number of participants also liked the fact that the focal point of the zoom did not move while using pressure-based input, which is usually not the case with pinch-to-zoom implementations.

I also developed a phone ringer application to evaluate the utility of the squeeze gesture. This applications plays a ringtone and goes into silent mode when the user squeezes the device. This application was tested by asking participants to keep the phone in their pocket or bag and once the app starts ringing, to reach into the bag or pocket, and squeeze the device. This squeeze action sends the device into silent mode and mimics the behavior of sending the caller to voicemail.

As is evident from the Likert-scale measures in Figure 3.19, participants frankly loved this application. Many wanted it on their phones immediately. P7 said, "*There was something satisfying about squeezing the phone and having the vibration stop instantly.*" P1 said, "*When can I get this? This is a really cool feature that I think would be very useful in all kinds of contexts.*"

One obvious utility of having pressure-based input is alternative input. Researchers have used similar input modalities for mimicking right-clicks [97], changing keyboard modes [42], and so on. I developed a custom smartphone keyboard that uses pressure input information to change letter case. Users can press harder to enter uppercase letters and press lighter to enter lowercase letters. Participants were presented with 5 phrases randomly selected from the MacKenzie and Soukoreff phrase set [149]. Forty percent of characters in each of these phrases were randomly converted to upper case. I also presented participants with a parallel app having the same interface and task, with the only difference being an absence of pressure-based input. Instead, the keyboard had a separate shift key for uppercasing letters.

Participants were asked to use this app while holding the device in one hand and interacting

with the index finger of the other hand. After the completion of tasks on each of the two keyboards, participants filled out the same Likert scales as for the Maps app. The difference in performance of the two keyboards was not as dramatic as it was in case of the Maps app. Nonparametric Wilcoxon signed-rank tests indicate that our Keyboard application required significantly less perceived workload on the *temporal* and *frustration* Likert scales ($p<.05$). Responses for effort showed a trend in favor of our Keyboard application (p=.07). Responses for the *mental*, *physical*, and *performance* scales were not significantly different.

The participants were divided on which keyboard they thought allowed them to type faster. For example, P4 said, "[UnderPressure felt like it took a lot less time to enter the text because I did not have to keep switching modes." Whereas, P7 said, "While using the pressure one I felt that typing was slower. I had to think more." A Shapiro-Wilk W test of normality indicates the *Time* measure differs significantly from normal ($W=0.95$, $p<.01$). However, a Kolmogorov's D test indicates the *Time* data does not depart significantly from lognormal (D=0.07, p=.15). Therefore, we log-transformed our Time measure before running a repeated measures ANOVA. Although the mean time taken for UnderPressure was a little less on average than the traditional Shift-based keyboard (14.98 s, $SD=5.29$ vs. 16.31 s, $SD=4.55$), the difference was not statistically significant ($F_{1,6}=2.69$, $p=.15$).

Although this custom keyboard shows that the pressure input can improve the user's productivity if they are switching between the cases on their keyboard, it doesn't utilize the full potential of GripSense. The actual task of typing is severely exacerbated when the user switches between hand postures [13], and considering GripSense detects the hand posture, there is an opportunity to lessen that exacerbation to some extent. In the next section I will discuss an approach to make a smartphone keyboard that adapts to the user's hand posture.

3.2.2.1.5 Discussion

Because UnderPressure uses the vibration motor to sense the pressure exerted, the motor

is triggered only when the user interacts with the touchscreen or in case of an infrequent event (*e.g.,* incoming voice call). I explicitly asked participants about the effects of vibration on their experience with the system. The majority of participants did not feel that their experience deteriorated due to this vibration. Significantly lower levels of frustration in the exit survey after using UnderPressure-based applications also support this finding.

I implemented UnderPressure on a Samsung Nexus S running the Android OS. Although the basic premise would remain the same, the pressure detection algorithms might need to be adjusted somewhat for different phones because of different physical characteristics. The variability of the sampling rate and resolution of different devices may also require algorithmic adjustments on some phones. Current inertial sensors present on commodity mobile devices are not high resolution and the techniques presented in this paper can benefit a great deal from improved resolution. The high performance exhibited by UnderPressure could be even better with improved sensor hardware in the future.

The use of the built-in motor to produce vibration means that almost half of the features are coming from a relatively high-frequency source. Hence, UnderPressure does not suffer from the usual limitations of inertial sensor-based techniques like the presence of external sources of vibration, *etc.* Although no formal study was conducted to measure the effects due to external vibrations, an informal test was conducted to estimate the efficacy of pressure sensing while sitting as well as walking; results were comparable for both postures.

As demonstrated by the results, the combination of touch-induced and motor-induced vibrations means that these techniques can be reliably implemented when the device is on a flat surface. Hence these algorithms can be ported to tablets as well, which are used relatively more on a desk when compared to a smart phone. Modern game controller manufacturers can also leverage these techniques with a simple software upgrade to add pressure sensitivity to their devices, as game controllers already have vibration motors and inertial sensors.

UnderPressure was evaluated for only three levels of pressure. Three levels were chosen to

make it easy for users to discern different pressure levels with acceptable levels of accuracy. I believe UnderPressure can actually infer more than three levels of pressure, amply demonstrated by the fact that even though the range of pressure applied by different participants was different, the system maintained high levels of accuracy. That said, a more continuous regression to pressure is possible and algorithms can be built on top of this work that have more than just three quantized levels of pressure.

3.2.2.2 Customized Touch: Adaptive Keyboard

In comparison to traditional desktop keyboards, text entry on touchscreen mobile devices is more challenging. Apart from being very small as compared to their desktop counterparts, these devices are used in very different contexts and situations. Moreover, there are situations where the user's dominant hand is occupied (*e.g.,* , supporting themselves with a grab-handle while standing on a moving bus) and the device is operated with the non-dominant hand's thumb. Research has shown that mobile device hand postures can significantly affect finger and thumb pointing performance [237], but such information has not been used for improving text entry, which required numerous rapid, accurate strikes and is a relatively high-intensity, if familiar, task.

In this section, I discuss *ContextType*, a smartphone keyboard that uses GripSense to infer the user's hand posture to improve text entry on mobile touch screen devices. ContextType supports typing with four hand postures: two thumbs, just the left thumb, just the right thumb, and either of the two index fingers. ContextType switches between underlying touch-models and keymaps based on the GripSense's hand posture inference (without changing the visual layout of the keyboard). Once the posture is inferred, ContextType combines a user's posture-specific touch-pattern information with a language model to classify the user's touch event as a pressed key, ultimately making text entry more accurate.

To design and build ContextType, I first collected touch screen typing data from 16 participants in all four hand postures. Based on this data, I built touch-based key-press

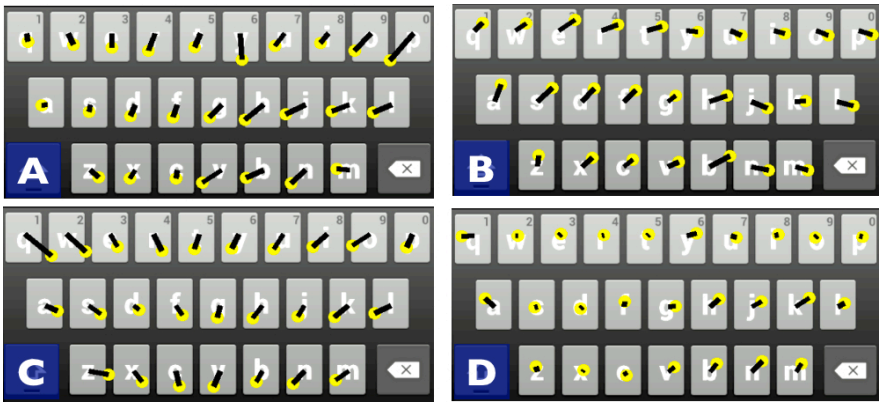

Figure 3.20: Tap patterns for different postures for a participant. *(A)* Left Thumb, *(B)* Right Thumb, *(C)* Index Finger, *(D)* Two Thumbs. The yellow spot is the touch centroid for each key an the line shows the drift from the visual key center.

classification models (one for each hand posture), personalized for each user. The final ContextType prototype is a composite of these personalized touch models and a 5-gram language model. These models will be discussed in detail later. This final smartphone keyboard was then evaluated with the same 16 participants. The control keyboard, to which ContextType was compared, also used personalized keyboards for each participant. The only difference was that the control keyboard did not take the hand posture information into account. The findings from the evaluation showed that ContextType decreased total text entry error rate by 20.6%. It was also found that inclusion of a language model does not have a significant improvement over the control condition.

3.2.2.2.1 Approach

ContextType combines three types of information to classify user's touch events as key pressures. It is informed by data about a user's hand posture, by a user's touch pattern, and also by letter probabilities from a language model. The algorithm develops different models for each hand posture. It uses GripSense to infer the hand postures and then personalizes the underlying keyboard layout for each user and their hand posture. Only the underlying keyboard layout is personalized according to the user's typing behavior (*i.e.,* the visual layout of the keyboard remains static). ContextType employs a constant,

diagonal covariance structure by computing a bivariate Gaussian distribution [64] for each key and centers each key at the centroid of predicted key pressed that are personalized for each user. Considering that touch behavior varies not only across participants but also across hand postures for the same participant, I generate separate keymaps for different hand postures for each participant. Figure 3.20 shows an example of the variance in touch behavior across postures for a participant. In the case of single thumbs, it is clear that the user tended to bias towards the side of the thumb because of its limited reach.

Language Model. ContextType includes a 5-gram letter model following the work of Goodman *et al.* [76]. The model was trained on the Brown corpus [124], consisting of American English from a variety of sources. ContextType uses the Modified Kneser-Kay method for probability smoothening, which has been successfully used by Chen *et al.* [38] in language modeling for soft keyboards. The validity and effectiveness of the language model was confirmed in a small stufy (6 participants), similar in apparatus and design to the final evaluation (described later). In this study, the participants were only required to complete 40 phrases in any one of their preferred hand postures. This study compares a static non-adaptive keyboard to a language model-powered keyboard. There was a significant increase in words per minute for the language model ($F_{1,315}$=57.14, p<0.0001). The language model also resulted in decreased error rates (Wilcoxon signed-rank test: Z=-2717.5,p<.0001).

Combining Touch and Language Models. ContextType combines the touch and language models by calculating the probabilities for eack key. The most likely intended key, k_i*, is given by:

$$k_i* = argmax_{ki} p_L(k_i|h).p_T(k_i|l)$$

where p_L is the language model probability, p_T is the touch model probability, k_i is the probability for each key, h is the language model history (last 4 entered characters in case of 5-gram letter model), and $l \in R^2$ is an x and y coordinate pair denoting the last touch

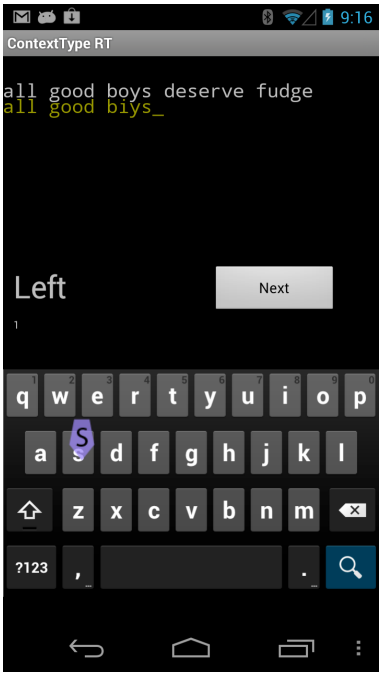

Figure 3.21: Test interface with the current posture and keystroke feedback. The data collection interface did not have the keystroke feedback.

location on the screen.

3.2.2.2.2 Evaluation

The touch models were built based on typing data collected from 16 participants (9 males, 7 females) who each volunteered for a 45-minute session. All participants self-rated as expert computer users and intermediate to expert touch screen mobile device users. Participants were between 22 and 33 years of age (M = 27.38, SD = 2.7). I built a custom data collection application for Samsung Galaxy Nexus phone (Figure 3.21). The interface was designed in a way to capture the user's natural typing pattern. Thus, similar to WalkType, it did not inform users of their mistakes and the correct letter was always displayed for each key press. The interface also allowed a swipe from right-to-left to remove the last typed character. Participants were instructed to swipe when they felt that they had made an error. Noisy data was removed by filtering out taps that landed outside the Euclidean bounds of the intended key or its immediate neighbors. Once comfortable

with the interface, the participants were asked to enter 30 phrases in each of the four postures. The order of postures was randomized and the phrases from MacKenzie and Soukoreff's phrase set were used. Apart from these phrases, every fifth phrase was a randomly selected pangram from a list of 35 pangrams to ensure sufficient representation of all letters of English alphabet.

I sought to see whether the knowledge of a user's hand posture could be used to improve text entry performance. In addition, I also wanted to investigate the effect of the language model on the overall performance of ContextType.

Participants. The same 16 participants who participated in the data collection phase were used for a second session, lasting approximately 1 hour, to evaluate ContextType.

Apparatus. Participants used a similar interface to the one used during the data collection phase. This time, the entered text contained the actual key classification and visual keystroke feedback.

Procedure. The session began with an introduction to the modified interface and explanation of the task. For each condition, participants completed 40 phrases. The application instructed the user to change hand posture after every five phrases. In the bottom-left corner of the text area, the current phrase number and current expected hand posture were displayed. The hand postures were counterbalanced and selected randomly.

Design Analysis. The study was a within-subjects 2×2×4 factorial design. The factors and levels were:

- ***Interface:*** *ContextType*, *Control*.
- ***Language Model:*** *Yes*, *No*.
- ***Posture:*** *Left Thumb, Right Thumb, Index Finger, Two Thumbs*.

When ContextType was running, the keyboard was personalized by leveraging touch data collected for each user and each of his or her hand postures. In the *Control* condition, the touch data was not partitioned for each hand posture. Hence, the control condition, though not adaptive to hand posture, had a personalized keyboard. Presentation of conditions was

counterbalanced. With 40 phrases in each condition, participants entered 2×2×40 = 160 phrases each.

The main measures were speed, calculated as words per minute (WPM), and total error rate [210]. Total error rate is decomposed into corrected and uncorrected error rates. Corrected errors are the errors that are subsequently corrected by the user before moving on to the next phrase. Uncorrected errors are those that are left in the transcribed phrase at the end of each trial. Also, the participants were asked to rate which of the conditions they preferred and why. The participants were not aware which condition was the current one to prevent bias.

For WPM, the results are from a mixed-effects model analysis of variance. For error rates, the nonparametric Aligned Rank Transform procedure was used [234]. The analysis used nonparametric analysis for error rates because error rates skew towards zero and violate normality. All pairwise comparisons were protected against Type I error using Holm's sequential Bonferroni procedure

3.2.2.2.3 Results

Speed. ContextType did not have a detectable difference in speed (27.5 WPM, *SD*=6.6 *vs*. 26.0 WPM, *SD*=6.2; $F_{1,2496}$=1.27, *n.s.*). However, *Posture* had a significant effect on WPM ($F_{1,2496}$=92.06, p<.0001). This was expected because participants generally preferred some posture to another. Post hoc pairwise comparisons showed that all postures were significantly different and that two thumbs were fastest, followed by right thumb, index finger, and left thumb. Left thumb's lower performance was expected because it was the non-dominant thumb for all participants. There was no significant *ContextType*×*Posture* interaction. Finally, there was no detectable increase in speed due to the language model (26.5 WPM, *SD*=6.3 *vs*. 27.0 WPM, *SD*=6.6; $F_{1,2496}$=2.20, *n.s.*).

Error Rate. Corrected error rates are subsumed in typing speed because correcting errors slows users down. Considering there was no detectable effect of ContextType on speed, I

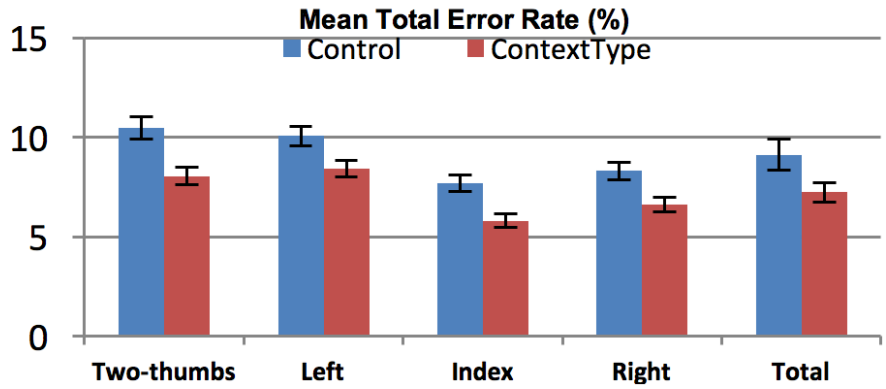

Figure 3.22: ContextType resulted in lower total error rate than the control condition for all the four hand postures. Error bars are the standard errors.

analyzed corrected error rates to investigate ContextType's performance further. Participants exhibited a marked and significant improvement in corrected error rate while using *ContextType* (4.86%, *SD*=2.4 *vs.* 6.49%, *SD*=4.4; $F_{1,2496}$=9.79, p<.002). *Language Model* also resulted in a trend toward reduced corrected error rates (F1,2496=3.09, p=.08).

There was no detectable difference in uncorrected error rate due to *ContextType* (2.38%, *SD*=1.58 *vs.* 2.63%, *SD*=2.35; $F_{1,2496}$=0.06, *n.s.*). No other factors had a significant effect on uncorrected error rate.

ContextType's effect on total error rate was also evaluated, which is the sum of corrected and uncorrected error rates. There was a significant effect of *ContextType* on total error rate ($F_{1,2496}$=10.87, p<.002). Compared to the control condition, total error rates decreased by 20.6% (Figure 3.22). In contrast to corrected error rate, the Language Model did not significantly affect total error rate. However, there was a significant *ContextType*×*Language Model* interaction ($F_{1,2496}$=3.94, p<.05). However, no post hoc pairwise comparisons were significant. As expected, there was a significant effect of Posture on total error rate ($F_{3,2496}$=4.97, p<.002). Post hoc pairwise comparisons showed that the left thumb was significantly less accurate than both the index finger ($F_{1,2496}$=12.12, p<.001) and the right thumb ($F_{1,2496}$=6.31, p<.02). This result was expected because left thumb was the non-dominant thumb for all 16 participants. The performance

with two thumbs was also found to be significantly less accurate than that of the index finger ($F_{1,2496}$=6.82, p<.01). Considering two thumbs had significantly higher WPM than the index finger, it suggests a speed and accuracy trade-off between the two postures.

Preference. Participants were also asked which of the two interfaces they preferred. Nine out of 16 participants chose ContextType. The remaining 7 participants did not perceive any performance difference. P7 said, *"[ContextType] was awesome! I did not have to look at the keyboard and the 'P' key felt much closer and accessible with my left hand"*.

3.2.2.2.4 Discussion

ContextType decreased corrected error rate significantly, but no significant effect on WPM was observed. Corrected error rate is generally correlated with WPM, which suggests that with more data, ContextType's improvement in typing speed might be detectable. Also, the decreased accuracy of ContextType for detecting posture (89.7%) might not be an impediment to its performance. Anecdotally, we analyzed results for one participant and found that ContextType primarily confused her index finger and right thumb. Upon further analysis I found that her typing pattern for index finger and right thumb were similar, thereby producing similar touch models. Our posture detection system does not require any calibration; hence posture-specific keyboard touch-models can be refined over continued usage without any user intervention. Although different language model implementations could produce significant improvements, ContextType's language model did improve performance over a static keyboard. It seems that the language model's benefit is largely negated in the presence of a personalized touch model.

3.3 User Attention

User attention is a very significant contextual factor that affects people's mobile device usage. As we use these devices in varied environments, our degree of attention varies as well. For example, if one is check email on their phones while walking, one must put their

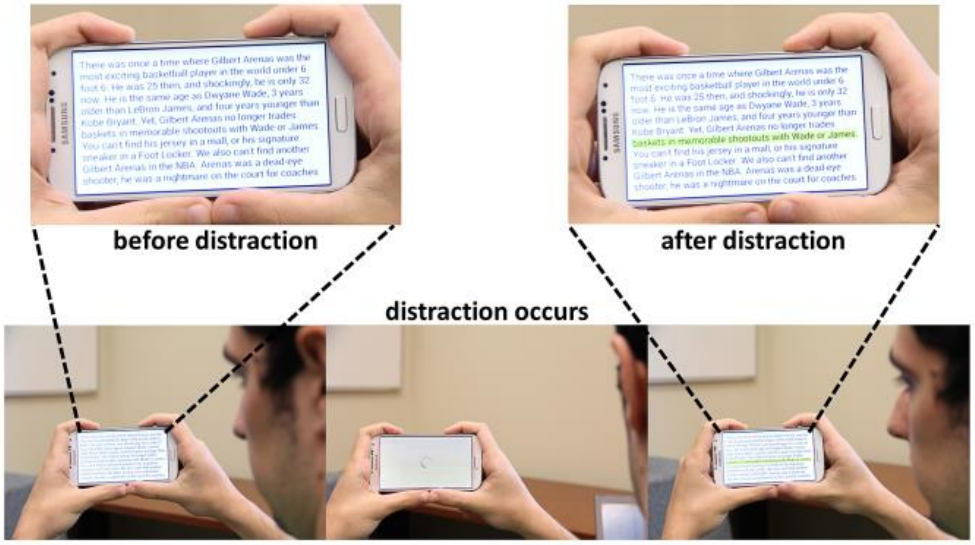

Figure 3.23: SwitchBack highlights where the user was last looking in a body of text before they turned away to handle a distraction.

phone away and break their attention from the device while crossing the street. Maintaining awareness of your surroundings mean that the level of attention to the task on the mobile device varies. However, once again our devices are not aware of these changes. In the aforementioned scenario, it is very likely that when the pedestrians return to their mobile task, there will be a startup cost to regain track of their progress. Context-switching incurs a startup cost that can accumulate to the point where users' comprehension is negatively affected [158].

In this section I will briefly discuss *SwitchBack* [151], a system that uses the mobile device's front facing camera to detect the user's gaze, infer what content on the mobile device screen is the user focusing on, and then remind the user of this inference when they come back to complete their task. It first determines whether the user is looking at the screen. If the user is looking and the phone is displaying text, SwitchBack detects saccades due to quick, horizontal jumps between lines as a tool to determine when the user moves to a new line. Keeping an estimated count of these line jumps gives a rough estimate of where the user is in the body of text. Once SwitchBack detects that the user has returned to the mobile task,

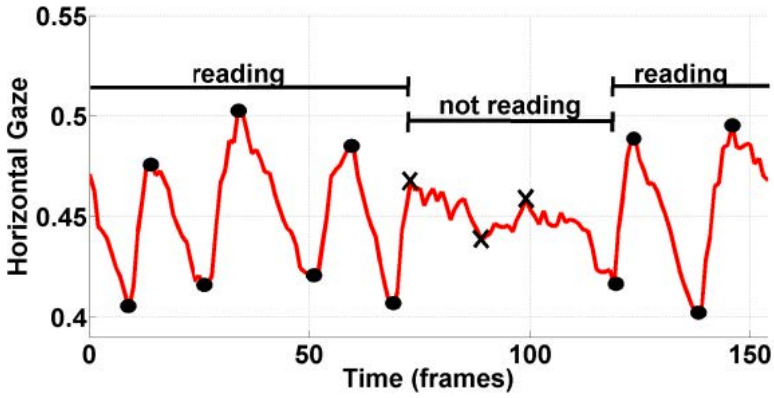

Figure 3.24: Horizontal gaze tracking. Peaks and troughs are detected throughout the signal but only indicate saccades when the amplitude of the signal is large. Low magnitude values correspond to when the user is looking to the right, whereas higher magnitude values correspond to looking left.

it guides the user's attention back by highlighting the appropriate portion of the screen (±1 inferred lines in case of text) (Figure 3.23).

To detect saccades, SwitchBack tries to detect the user's pupil by finding the darkest portion of their eye. These estimates are then passed through a first-order infinite impulse response (IIR) filter with a cut-off frequency of roughly 2.5 Hz; this is a low-complexity low-pass filter that suppresses high-frequency noise. Figure 3.24 plots the estimated horizontal location of a user's pupil with respect to time as the user reads a body of text. Low magnitude values correspond to the user looking to the right, whereas higher values indicate a gaze in the left direction. The signal jumps whenever the user's gaze changes drastically and a saccade occurs. A detected jump is only counted as a saccade if the gaze location changes significantly afterwards. This helps in making sure that the system is prone to noise appearing in flat signals (especially true if the user fixates on a portion of text for a long time). In Figure 3.24, the local optima marked with circles are considered saccades by the algorithm, while the crosses mark some of the local optima that are rejected by the algorithm, because the signal exhibits less deviation during those times.

In an evaluation with 17 participants, SwitchBack was able to estimate how many lines the user has read in a body of text with an error of 3.9%. This is yet another demonstration

that the on-device sensors on the mobile devices can help the device in becoming far more aware of their usage environment than the current state-of-the-art. Furthermore, once the device is aware of the user's level of attention and it helps the user in resuming their task, SwitchBack improved reading speed by 7.7% (roughly 19 words per minute) when compared to a control condition (*i.e.,* absence of SwitchBack).

3.4 Consequences & Summary

Using mobile devices in a variety of environments can lead to situational impairments due to, among other sources, vibration, posture, and divided attention. In this chapter I have discussed a number of efforts that use the on-device sensors (such as accelerometer, gyroscope, touchscreen, and camera) to sense and accommodate for some of these situational impairments.

I discussed a motion adaptive keyboard, *WalkType*, that leverages the on-device accelerometer to compensate for vibrations and extraneous movements caused by walking. It increased typing speeds from 28.3 WPM to 31.3 WPM, and also reduced the number of uncorrected errors from 10.5% to 5.8% while the participants were walking. Then I discussed *GripSense*, a system that leverages on-device capabilities, such as touchscreen and gyroscope, to infer the user's hand posture with a combined accuracy of over 87%. *UnderPressure* uses gyroscope and vibration motors to infer the amount of pressure exerted on the device and can differentiate between three levels of pressure with an accuracy of 95.1%. I then went on to discuss *ContextType*, a keyboard that builds on top of GripSense to develop hand posture specific keymaps for each other and improving typing performance by over 20%. Finally, I discussed *SwitchBack* that improves reading speed by over 7% by reminding the users about their progress while reading a body of text in presence of distractions. Overall, this chapter shows that there is an incredible opportunity to use the on-device sensors to make our device more aware of the environments in which they are being used. I have taken special care to make sure that with each exploration I also study the benefit of these newfound awareness on the user

experience. In every case, the increased awareness led to more efficient use of the device. However, these technologies do not come without their challenges.

First, using on-device sensors can be power consuming. For example, one cannot leave the vibration motor running the whole time to sense the amount of pressure exerted. Not only will it lead to degraded user experience, it will also drain the battery. The applications need to be intelligent in using these capabilities.

Second, the inferences from the on-device sensors can be slow. This means that the applications using these inferences need to adjust accordingly. For example, in case of GripSense, one can imagine that an application might prefer drastically changing the visual layout on the basis of user's hand posture. However, GripSense takes 5 interaction steps before inferring the user's hand posture. Therefore this information cannot be used to drastically change the visible layout of the phone. A visual change should be immediate and react quickly to the user's state change. The reason why ContextType is successfully able to use GripSense is that it does not affect a visual change to the interface. The change happens underneath the visual layer and even if there is a delay or even a mis-classification, the user experience does not degrade significantly. It is important that developers and designers keep these challenges in mind while using the technique described in this chapter and using the on-device sensors, in general.

This chapter demonstrates that the on-device sensors can be used as substitutes for dedicated sensors that detect a user's posture and attention information. The extended aim of this dissertation is to study the role of multiple sensors when a single sensor falls short. GripSense highlights the use of multiple sensors in detecting the user's hand grip and the final inference is far more robust when the inferences from gyroscope, accelerometer, and touchscreen are combined. However, a more rigorous deployment of these applications is necessary to study the limitations of the current approach in more detail.

Multiple applications in this chapter also highlight another dissertation question: "How do these generic sensors cope with different performance requirements". For instance,

WalkType detected the user's gait almost instantaneously, however GripSense took around 5 interaction steps to reliably infer the user's hand posture. The adaptive keyboards, however, showed improved typing accuracy and speed in both cases. This shows that sensors and applications were able to cope with different performance requirements. Overall, the applications need to be sensitive to the inference capabilities of the indirect sensors.

Chapter 4

Multi-Device Interaction

It is common to have multiple computing devices co-located in the same environment. In a number of these situations, it is useful for these devices to communicate with each other. Photo sharing is a canonical example, where a user may want to take a picture on their smartphone and share it with another smartphone, any number of computers and tablets, or even a local projector. Although the networking and data transfer aspects are largely solved using technologies such as Wi-Fi, Bluetooth, and NFC, users still need simple and natural ways to indicate the intent for creating logical connections between devices.

The radio signal strength can be used to identify proximate devices, but in today's device-dense environment, users must select which devices within wireless proximity should belong to a particular group. For instance, if the user wants to send a photo from their mobile phone to another person's computer through Bluetooth, they need to identify the computer's name from a list of nearby devices in order to pair with it.

A number of techniques [95, 96, 98, 101, 154, 183, 199] have been developed to establish a connection between devices vased on "*context proximity*" [98], but these techniques do not scale well with increasing numbers of devices. Most of these techniques assume one or two devices per user, but it is increasingly common for users to have multiple devices for

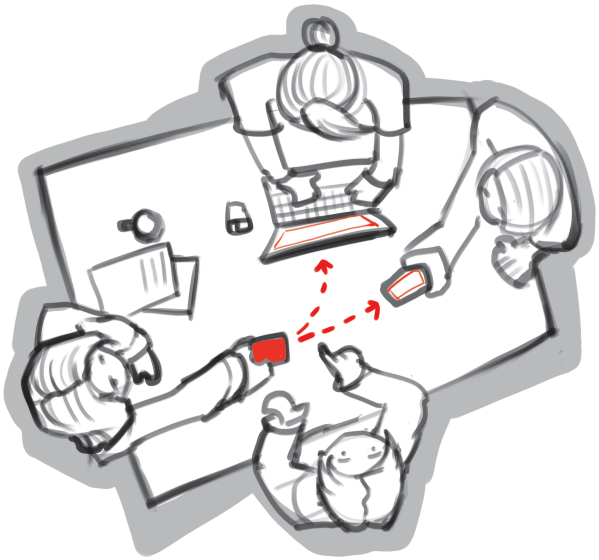

Figure 4.1: Users interact with multiple devices on a surface

varying purposes. Thus, prior "pairing" techniques, which were suitable for a literal *pair* of devices, are inadequate for today's multi-device environments.

4.1 Devices Kept on the Same Surface

One common setting where devices and their users exchange information is around a table or other shared surface. For example, colleagues collaborate around a conference table, friends share photos around a coffee table, or a single user has their phone, tablet, laptop, *etc.* . on a single desk. Hence, this commonly available flat surface provides an expedient medium for interacting across multiple devices within a bounded context (the shared surface). In order to enable multi-device interaction on a shared surface, three attributes need to be sensed: (1) the presence of multiple devices on the surface, (2) their relative placement on the surface, and (3) the gestures the user performs to interact between devices. Although past work has relied on hardware such as cameras and IR-LED arrays [30, 50, 106, 152, 231], similar capabilities can be enabled through simple on-device inertial and acoustic sensing. In this section, I discuss *SurfaceLink* (Figure 4.1),

a system that leverages inertial and acoustic sensing to enable multi-device interaction on a shared surface. Using only the on-device sensors plus an optional 0.20 USD contact microphone plugged directly into the microphone jack, SurfaceLink detects the presence and placement of devices and gestures between them. Using sophisticated noise cancellation, the contact microphone becomes unnecessary, and SurfaceLink can be a completely software-based solution.

SurfaceLink detects the presence of all devices on the same surface by using the surface material's ability to conduct vibrations. To induce vibrations on the surface, the user can either knock on the surface manually or the device can vibrate using its built-in vibration motors. These vibrations are sensed through accelerometers and contact microphones. Any devices that are not on the shared surface will not be able to sense these vibration patterns, thus will not consider themselves part of the same group. The surface therefore serves as a "location limited channel" [15] for situations where users want groups that only include devices directly on the shared surface.

In addition, SurfaceLink turns most surfaces into an input medium where different types of single and multi-touch gestures can be performed to interact between devices. When a finger is dragged over a surface, it produces vibrations on the surface that can be sensed as gestures using microphones. This phenomenon has been previously demonstrated by Harrison and Hudson [85]. SurfaceLink extends ScratchInput by adding multi-touch, directionality, speed, and length to develop a richer set of gestures. SurfaceLink also detects the relative arrangement of devices on the surface by combining a user's surface gestures with acoustic stereo positioning.

I evaluated SurfaceLink in a controlled user study with 10 participants. Our findings showed that SurfaceLink detected device presence on the same surface with 97.7% accuracy and various surface gestures with an average accuracy of 90.3%. SurfaceLink also performed the relative localization of devices with 89.4% accuracy. Lastly, I qualitatively evaluated the usefulness of SurfaceLink and showed that SurfaceLink scales better to increasing numbers of devices than current multi-device interaction techniques.

The main contributions of this work are: (1) multiple techniques to enable rich ad-hoc multi-device interactions using the surface as an input medium; empirical results from an evaluation of SurfaceLink showing that it (2) robustly detects the presence of multiple devices on the same surface and establishes connections, (3) continuously tracks single- and multi-touch gestures between multiple devices by inferring their direction, length, speed, and touch modes, and (4) accurately detect the relative arrangement of devices.

4.1.1 Past Work

4.1.1.1 Vision-based Surface Interaction

For over 15 years researchers have emphasized that the surfaces around us can be used for interaction [190]. HoloWall [152] is one of the first systems that allows a user to interact with a glass wall. The system leverages IR-LEDs, a camera, and a projector. Similar to HoloWall, many recent techniques have used vision for making interaction with the surface more engaging. BonFire [106] uses a camera and two projectors to improve interaction between a laptop and a table's surface. Cuypers *et al.* [50] developed a technique to determine the position of a portable device on an interactive glass surface by projecting unique visual patterns and sensing the pattern received by the portable device's camera. BlueTable [231] connects multiple devices on a shared surface using vision-based handshaking. SideSight [30] uses IR-LEDs and vision for around-the-device interaction. All these systems, whether single- or multi-device, use computer vision to detect the devices and user-interaction with the surface. A vision-based system lacks portability and requires one or more cameras in the environment. These requirements make such a system unsuitable for impromptu interactions. In contrast, SurfaceLink uses lightweight inertial and acoustic sensing to detect devices and enable user-interaction. This approach allows for much more impromptu multi-device interactions.

4.1.1.2 Detecting Contextual Presence

Apart from the vision techniques mentioned above, others have proposed techniques for detecting the presence of multiple devices that shared some context with each other. Smart-Its-Friends [98] called this concept *"context proximity"*. The authors described how artifacts established connection with each other by detecting a shared phenomenon in their relative contexts. For example, two devices shared data when a user shook both of the devices simultaneously. This technique of detecting synchronous events has been explored by a number of researchers. Ideas such as bumping two devices together [154] or making contact with touch screens at the same time [183] provide a tangible way of initiating connections between two physically disconnected devices. PhoneTouch [199] detects co-occurrence of device vibration and surface touch to facilitate interaction between an interactive surface and a mobile device. Hutama *et al.* [102] used tilt correlations to facilitate interaction between multiple devices. All these techniques have a scalability problem as they either do not work for more than two devices, or assume a limit of about one user per device. Stitching [95, 96] provides a very interesting way of connecting multiple devices together. The user draws a line with their finger from the touchscreen of the sending device to the touchscreen of the receiving device. This is a very intuitive action for connecting two devices, but it requires both devices to have a touchscreen and is less intuitive for connecting more than two devices. SurfaceLink avoids the scalability problem by leveraging the mutually sensed vibrations of a shared surface to allow any number of devices to be easily connected.

4.1.1.3 Device Arrangement

Detecting where the devices are kept in the interaction environment is relatively straightforward in the earlier mentioned vision-based techniques. The main disadvantages of such techniques are their relatively demanding hardware requirements. These technologies cannot be used for impromptu interactions because it is not common to have calibrated cameras set up in the environment. Dearman *et al.* [54] developed a

lightweight system where a group of mobile devices used their backside camera to infer their relative orientation. Lucero *et al.* [146] explored how radio tracking technology can be used to determine devices' relative positions, but this approach fails when the devices are densely packed. Similar to SurfaceLink, Kortuem *et al.* [120] used ultrasonic audio signals for relative localization of devices, but they used external hardware and were limited to only detecting whether the device is *to the left, to the right, approaching,* or *moving away* from first device. SurfaceLink, in contrast, combines surface gestures and stereo acoustic sensing to infer a multi-device arrangement on a surface in both x and y directions.

4.1.1.4 Surface Gestures using Acoustic Sensing

Prior to SurfaceLink, others have proposed techniques for enabling interactions with a surface using acoustic sensing. Paradiso and Checka did some of the early work in this area [172]. They proposed a system that tracked a user's knock on a glass wall by measuring the time delay of arrival (TDOA) of vibrations at 4 different contact piezoelectric pickups. The work by Crevoisier and Polotti [48] was one of the earlier works that explored interaction with day-to-day physical artifacts through sensing the surface vibrations. Scratch Input [85] demonstrated that when a user drags their nails on a textured surface, the resultant vibrations could be used to convert the surface into an ad-hoc interactive surface. ScratchInput just scratched the surface of possibilities in this area, and SurfaceLink builds on this work to additionally provide single- and multi-touch gestures with varying speed and length. The gestures can be tracked in real-time and because SurfaceLink performs spectral analysis on the acoustic signal, it performs well even when the signal-to-noise ratio is low. Hence, the user no longer is limited to scratching with just the fingernails. The user can interact using any part of the hand. SurfaceLink is also first to explore how these gestures can be used in a multi-device environment and presents a set of gestures and enabling technologies for effectively interacting among multiple devices.

Seniuk and Blostein [202] explored how the surface vibrations generated by pens on a

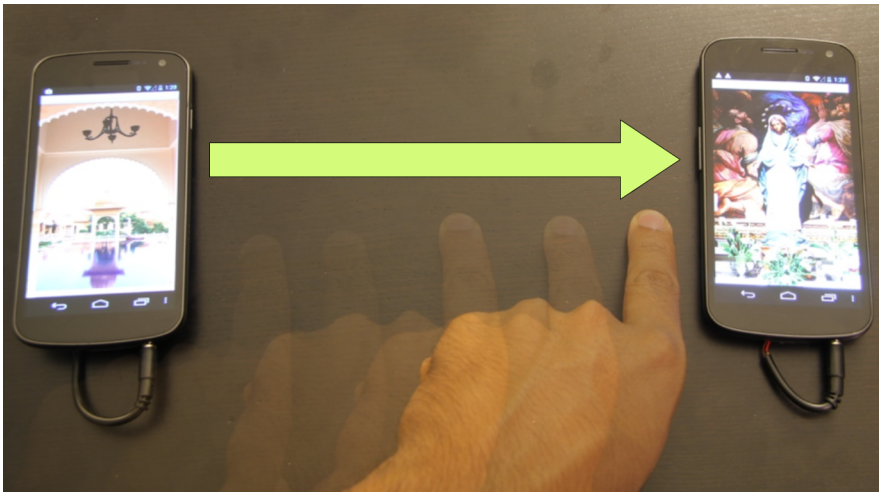

Figure 4.2: User interacting with to mobile devices on a shared surface using on-deive inertial and acoustic sensing.

known textured surface could be used for recognizing a fixed dictionary of 26 words written by the user. Murray-Smith *et al.* [162] also used such surface vibrations to develop a handheld interaction device that could be controlled by tapping, scratching, or rubbing the surface. In a similar effort, Harrison*et al.* [87] demonstrated how structured patterns of physical notches could be used as acoustic barcodes when swiped with a hard object like a fingernail or a smartphone. TapSense [86] and similar work by Lopes *et al.* [140] leveraged the contact microphone on a touch screen to detect different touch modes while a user tapped on the screen. SurfaceLink applies this same approach, and further extends the gesture set. To the best of my knowledge, SurfaceLink is the first work to demonstrate and evaluate gesture length, speed, direction, multi-touch, and application of these gestures in a multi-device environment.

4.1.2 System Design

SurfaceLink enables end-to-end fluid multi-device interactions. It first detects contextually proximate devices, then enables a rich set of single- and multi-touch gestures, and combines them with stereo acoustic sensing to determine the relative positions of the devices. A contextual group of devices present on the same surface is

created. Then the user can interact with these devices through gestures on the surface. SurfaceLink enables a number of on-surface gestures including single-finger directional swipes 4.2, multi-finger pinch and expand, and grouping gestures. It also enables robust detection of the length and speed of these gestures, which can be used for continuous, fluid interaction between devices. In order to provide the system with a much better understanding of the arrangement of devices, SurfaceLink combines stereo positioning with user gesture data to accurately detect the relative positions of devices in a 2-dimensional space.

SurfaceLink leverages the shared surface as a communication channel between the devices that are placed on it. It uses a combination of the on-device accelerometer, vibration motor, speakers, microphones, and off-device contact microphones for establishing connections and interactions between the devices.

4.1.2.1 Detecting Devices on the Same Surface

SurfaceLink uses the fact that hard, flat surfaces conduct vibrations well. These vibrations can either be user-induced by knocking on the surface, or device-induced using a vibration motor. These vibrations travel through the dense material of the surface and are sensed by the device sensors. Harrison *et al.* [85] and Kunze *et al.* [126] demonstrated techniques to detect the nature of the surface on which a device is kept. In contrast, SurfaceLink detects if the devices are on the same surface, irrespective of the material. For example, SurfaceLink can distinguish between whether devices A and B are kept on the same or different tables, whereas earlier techniques detected if a device was kept on a metal or wooden table.

4.1.2.1.1 Using User Induced Vibrations

When a user knocks on a surface, it generates strong vibrations that travel through the dense material of the surface. These vibrations can be sensed by the on-device accelerometers. Initially, when a user wants to connect to other devices on the same surface, the user's device acts as a host device and uses GPS and Wi-Fi information to find

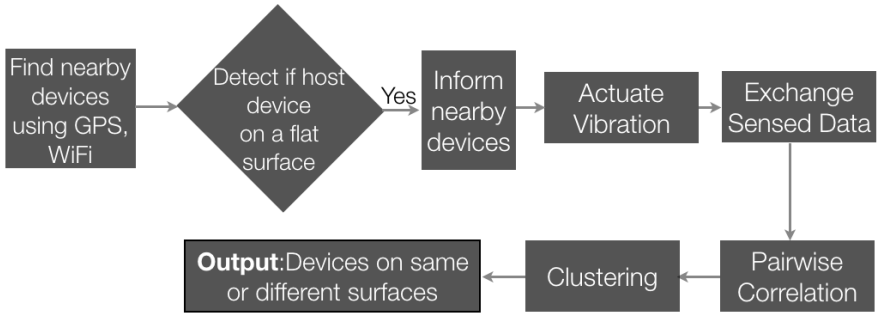

Figure 4.3: Block diagram of major components of SurfaceLink's detection of devices on the same surface. When sensing user-induced vibrations, inertial sensor data is used, and when sensing device-induced vibrations, the microphone is used.

the nearby devices. Once the device has a list of nearby devices, it instructs them to start sampling their accelerometers (Figure 4.3). It then instructs the user to knock on the surface in any user-defined pattern and samples its accelerometer at 100 Hz. The devices share their observed accelerometer data. A device might experience a number of vibrations in a busy environment but most of these vibrations are relatively low frequency when compared to those generated by the user's knocks. Hence, the accelerometer data of all devices is passed through a high-pass filter. Then a pairwise cross correlation is performed between these data samples. The devices are then clustered into two sets (*on same surface*, or *not*) on the basis of their cross-correlations. I use the Single Linkage algorithm that uses the smallest distance between objects in different groups to cluster them.

4.1.2.1.2 Using Device Induced Vibrations

In a number of cases, interface designers might not want to require a user to generate the stimulus by knocking on a surface to pair devices. In such cases, I can use the built-in vibration motor to generate the stimulus. The inertial sensors on most mobile devices cannot detect device-induced vibrations because they do not have the sensitivity and resolution required. Hence SurfaceLink uses a microphone to pick up these subtle vibrations. These vibrations are usually coupled to the air and can be heard by a microphone on some other surface as well. In order to pick up these subtle surface

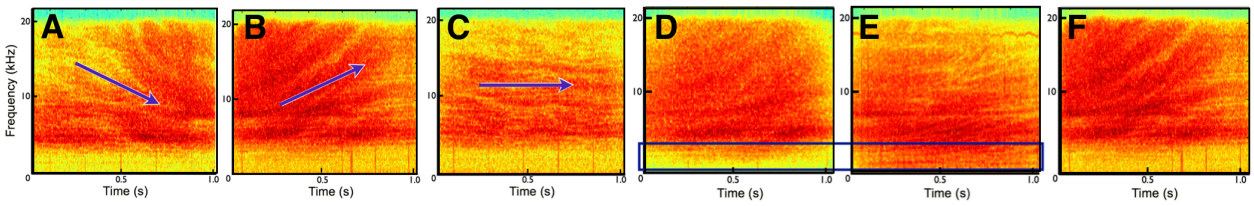

Figure 4.4: Audio spectrograms from various surface gestures. (A) Swipe towards the device. (B) Swipe away from the device. (C) Vibrations experienced by the non-participating device. (D) Gesture performed with fingertip. (E) Gesture performed with the fingernail. (F) Gesture performed with fist.

vibrations, it requires either noise cancellation using two microphones on the device (*i.e.,* one touching the surface and one in the opposite direction), or a low-cost external contact microphone with a resonant frequency of 6.3 kHz. The device uses on-device inertial sensors to automatically detect if it is placed on a stable, flat surface. In the same way as mentioned above for the user-induced case, the device obtains a list of nearby devices. These clients are then informed to sample their microphones at 44.1 kHz. When a device vibrates on a hard, flat surface it periodically hits the surface, thereby generating a low frequency sound. The system performs noise cancellation by subtracting the audio from the microphone on the top from the microphone touching the surface. When using a contact microphone, the microphone couples strongly to the surface and only records the surface vibrations, hence no filtering is needed. The devices then share their data and a pairwise cross-correlation and clustering is done as described previously.

In some cases, where there are a number of devices on a relatively large table, the host device can be far from some potential client devices. This often leads to situations where the devices are unable to reliably experience the induced vibrations. In such a situation, the system assumes that these far-away devices would still be relatively near to some of the other devices and employs a "vibration daisy chaining" technique where identified devices induce new vibrations to extend the reach of the device network.

4.1.2.2 Detecting Surface Gestures

SurfaceLink uses the concept that a finger/hand dragged over most hard surfaces produces measurable vibrations. These vibrations can be detected using the on-device microphone or an off-device contact microphone.

4.1.2.2.1 Gesture Type

I observed that the vibrations generated due to dragging of finger on a surface have different characteristics in different directions. One of the differences is that in many cases they are louder when near the device and softer when far. The difference in swipe direction also results in distinctly different patterns in the frequency domain. As a user's finger comes near a device, there is a decrease in the resonant frequency (Figure 4.4A), and the opposite effect is observed when the finger moves away from the device (Figure 4.4B).

A swipe gesture may be performed between two devices, but in a number of situations, more than two devices might be present on the surface. The devices not involved in the gesture would also hear the vibrations on their microphones. Hence there is a need to disambiguate between participating and non-participating devices. In the case of a unidirectional swipe gesture between two devices, a non-participating third device can experience one of three possibilities. It will observe vibrations similar to the two participating devices if it is right next to one of the participating devices. In this case, the overall spectral energy of the vibrations will be much higher for the participating devices. If the non-participating device is not near to either of the participating devices, it might observe similar energy levels, but now the swipe is neither going towards nor away from it, and hence it will observe vibrations similar to those shown in Figure 4.4C (*i.e.,* no change in resonant frequency over time). SurfaceLink also supports two multi-finger gestures: *pinch* and *expand*. In the pinch gesture (Figure 4.5, Left), the user starts the swipe near each device and draws them together toward the middle. This gesture can be used to connect two devices together. In the case of such a gesture, both devices experience a swipe away from them (Figure 4.4B). The complimentary gesture, expand

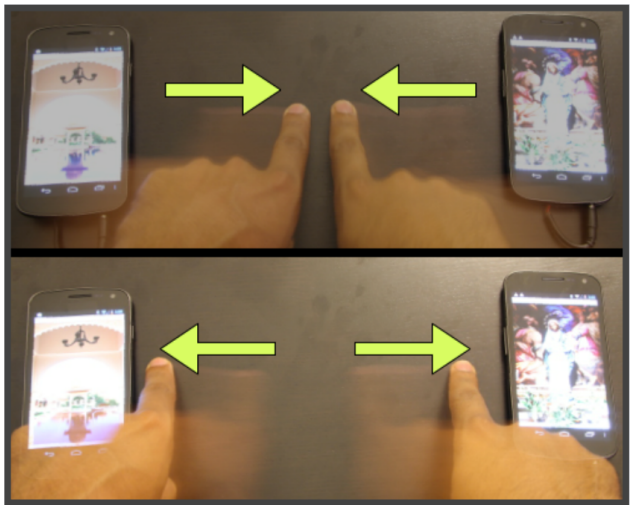

Figure 4.5: (Left) Pinch Gesture, (Right) Expand Gesture

(Figure 4.5, Right) generates the equivalent of swiping toward both devices (Figure 4.4A) and can be used for disconnecting two devices.

The speed of a gesture on a surface is directly correlated to the amplitude of the resonant frequency. In addition, the duration of all gestures are bounded since they are mostly performed between two devices. Hence, if a user does a fast swipe, the gesture accelerates and decelerates quickly. Therefore, in addition to the higher amplitude, the slope of increase and decrease in amplitude is also steeper. The combination of these two phenomena can be used to differentiate between different gesture speeds. In order to make it easy for the user to discern different speed levels, SurfaceLink supports only two speeds: slow and fast

4.1.2.2.2 Gesture Length

Considering that SurfaceLink can estimate the speed of a gesture and the length of a gesture is bounded by the distance between two devices, I can also detect the length of a gesture. A smaller gesture can be compared to a longer gesture of same speed as both will have the same amplitude and slope but different durations. SurfaceLink supports three gesture lengths: full, half, and quarter length.

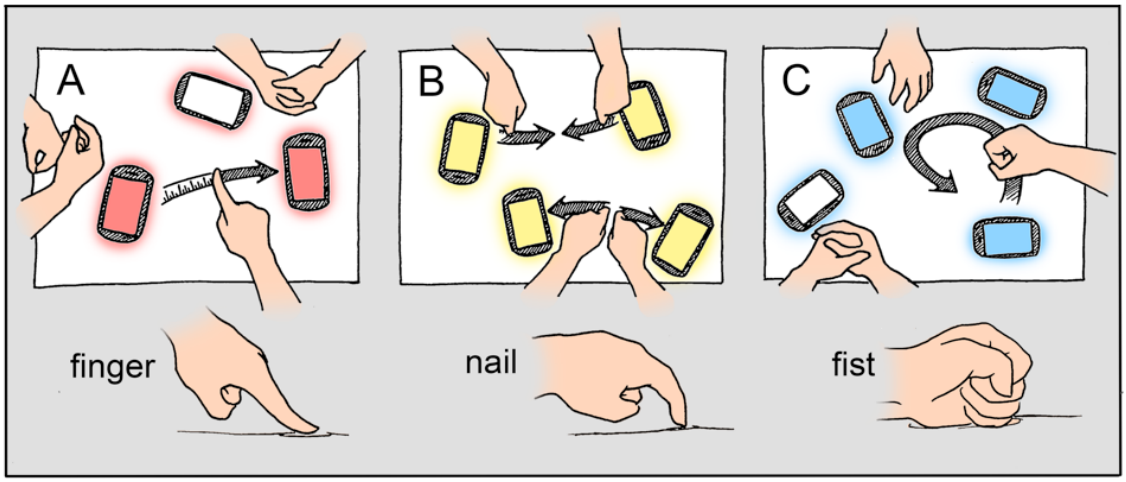

Figure 4.6: Devices on each table are detected by SurfaceLink and a local network between them is created. Users can then interact with these devices through a number SurfaceLink-enabled gestures and touch-modes.

4.1.2.2.3 Touch Mode

SurfaceLink uses the fact that different finger locations generate different vibrations when dragged across a surface. SurfaceLink employs the techniques demonstrated by Harrison *et al.* [86] to support three touch modes (Figure 4.6): fingertip, fingernail, and fist. The nail, being harder than the fingertip, produces a very different frequency profile (Figure 4.4D and 4.4E). The differentiation between fingertip and fist (Figure 4.4F) is slightly different. The fingertip and fist are made up of the same material, so the frequency response for the two modes is not different. However, because the area of contact is much larger in the case of the fist, it creates a higher amplitude sound.

4.1.2.2.4 Gesture Shape

ScratchInput [85] demonstrated that analysis of peak counts and amplitude variation could be used to infer a fixed dictionary of gesture shapes, such as a line, circle, triangle, and square. I employ similar techniques, albeit using the spectral information and performing pattern matching.

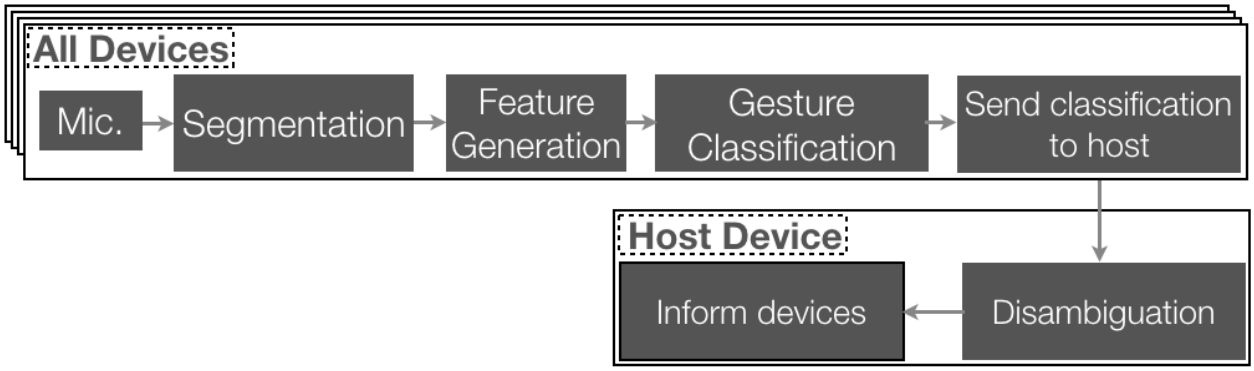

Figure 4.7: Block diagram of major components of SurfaceLink's gesture detection module. Pattern matching results from all devices are sent to the host device for disambiguation.

4.1.2.2.5 Implementation

Every device on the surface records audio data at 44.1 kHz. The data is then segmented into individual gestures. This segmentation is done by thresholding the audio data between 5 kHz and 15 kHz. I did not formally evaluate the segmentation but in our informal evaluation with 3 participants, there was virtually no segmentation error because the contact microphone couples very well to the surface and is immune to ambient noise. The segmented audio data is then used to generate machine learning features before going through gesture classification (Figure 4.7). Each device individually infers the gesture, and then shares the classification result with the host device. SurfaceLink classifies each gesture into four different gesture properties:

- ***Gesture Class:*** Away, Toward, Non-Participating, Pinch, Expand, or Fast Swipe.
- ***Gesture Length:*** Quarter, Half, or Full.
- ***Touch Mode:*** Fingertip, Fingernail or Fist.
- ***Gesture Shape:*** Line, Polygon, Triangle, Circle, or Semi-Circle.

In order to generate the features, I produce the magnitude spectrogram of the audio signal using a 1024-point FFT with a 100 sample Hamming window. In order to reduce the number of features, the |FFT| data over the duration of the gesture was down-sampled into 6 frequency bins and 10 time frames using median filtering. This forms a 6x10 matrix representing the first 60 features used for classification. Next, in order to capture

the temporal variation for each gesture, I band-pass filtered (5 kHz to 15 kHz) the temporal data and then down-sampled into 10 time windows. Thereby providing 10 new features. The last 2 features are the total energy in the first and second half of the gesture. These 72 features are used to inform a kNN classifier (k = 2) on each device. I have separate kNN classifiers for each gesture property (*Gesture Class, Gesture Length, Touch Mode,* and *Gesture Shape*). It helps in keeping the number of training samples to a minimum. A detailed analysis of the effect of training size on the performance of the system is provided in the Results section.

Once each device makes a decision for each gesture, it sends the classification result, classification confidence, and the total observed energy for the gesture to the host device. The inverse of normalized distance between the observation and nearest neighbor is used as the classification confidence. For example, if the normalized kNN distance between the observation and nearest neighbor on device A is 10, and on device B is 20, then their respective classification confidences are 0.1 and 0.05. The host device then checks if all the devices agree on the classification. In case there is ambiguity, the host device checks the confidence metric for each device and selects the decision of the device with the highest confidence. In cases where multiple devices predict that they were participants in a gesture, the device with higher total observed energy is selected.

4.1.2.3 Detecting Device Arrangement

SurfaceLink also infers device arrangement that can be used to enable placement-aware interactions. It leverages the knowledge of the shape of the gesture to infer the arrangement of devices. For example, in Figure 4.8 Left, the circular gesture began and ended at Device A. As a result, different devices on the table observed a peak in the vibrations at different times and the non-participating device E, as expected, observed significantly lower vibrations. From these signals, the ordering and placement of the devices can therefore be inferred. To do so, the system places the objects on the edge of the drawn shape (*e.g.,* a circle as shown in Figure 4.8, Left). To determine the distances between the devices, the system uses the time intervals between the peaks in the audio

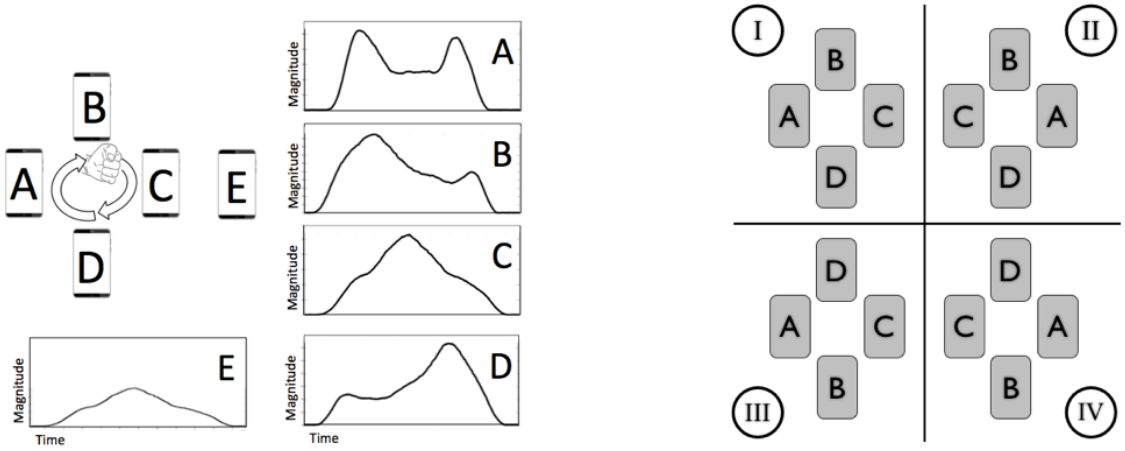

Figure 4.8: ***(Left)*** **The timestamp of the audio peak for each device informs the order of the devices.*(Right)* Four possible arrangements of devices**

received by each device. For example, if B receives a peak closer in time to A than C, then the system infers that B is physically closer to A on the edge of the circle. Also, if device E observes an attenuated version of the peak observed by C, but with the peak at the same time, then the system infers that C and E are at the same angle on the circle, but at different radii (*i.e.,* , E is farther from the gesture).

This inference system has one limitation: the system would come back with the order of devices as A-B-C-D. The actual arrangement can be any of the four possible mirror images (Figure 4.8, Right). In order to reduce the ambiguity, SurfaceLink limits the user to perform clockwise gestures when devices are in a 2-dimensional arrangement. By adding this constraint, the system can eliminate arrangements shown in Figure 4.8, Right (III) and (IV). In order to reduce the remaining ambiguity, SurfaceLink employs an acoustic stereo positioning technique to calculate the right/left orientation of one device with respect to another. For example, in Figure 4.8 Left, if Device A infers that Device C is to its right, then the only possible arrangement will be Figure 4.8 Right (I). In cases where devices are arranged in a single line (1-dimensional arrangement), there are only two mirror images. This ambiguity can be easily resolved by stereo positioning, hence there is no limitation on the user for gesture direction.

In order to determine whether a device is on the left or right side of another device, it emits

two ultrasonic tones of different frequencies from both left and right speakers. Ultrasonic tones have been used for data transfer [11] and proximity detection [215] as well; however, these have never been used for relative arrangement detection. I use an 18 kHz sine wave for the left speaker and 18.5 kHz for the right speaker. If the client device is on the right side, then the observed amplitude of the 18.5 kHz tone will be higher than that of the 18 kHz tone. The frequency response for most commodity speakers is not linear at such high frequencies, hence there might be a constant offset between the magnitudes of the two frequencies. Such an offset can be easily accounted for by calibrating using the microphone of the emitting device.

In the ideal case, the amplitude of the two tones can be compared instantaneously, but I observed that the difference in amplitude could vary over time. This leads to some noisy results, hence SurfaceLink tracks the amplitude difference for 3 seconds and passes it through a low pass filter before deciding on the orientation. After this decision, the system does not re-compute the orientation until the user moves the devices.

Such placement-aware systems have been studied earlier as well. Hinckley *et al.* [96] demonstrated the utility of such a system using touchscreens and the slope of a user's swipe while connecting devices. This knowledge of relative placement of devices can be used to facilitate more natural touch-screen interactions between devices. For example, if a user swipes from left to right on the touchscreen of device A, the system detects whether device B is on right or left. If B is on left, then it means that the gesture went from B to A and so on.

4.1.3 Evaluation & Results

The performance of SurfaceLink was evaluated in a controlled user study. Ten participants (6 males, 4 females) ranging in age from 21 to 32 (μ=26.9, σ=3.6) were recruited. All participants had more than 10 years of experience with computers and self-rated as intermediate to expert computer and smartphone users. The evaluation was performed using a custom developed application deployed on 4 Samsung Galaxy Nexus

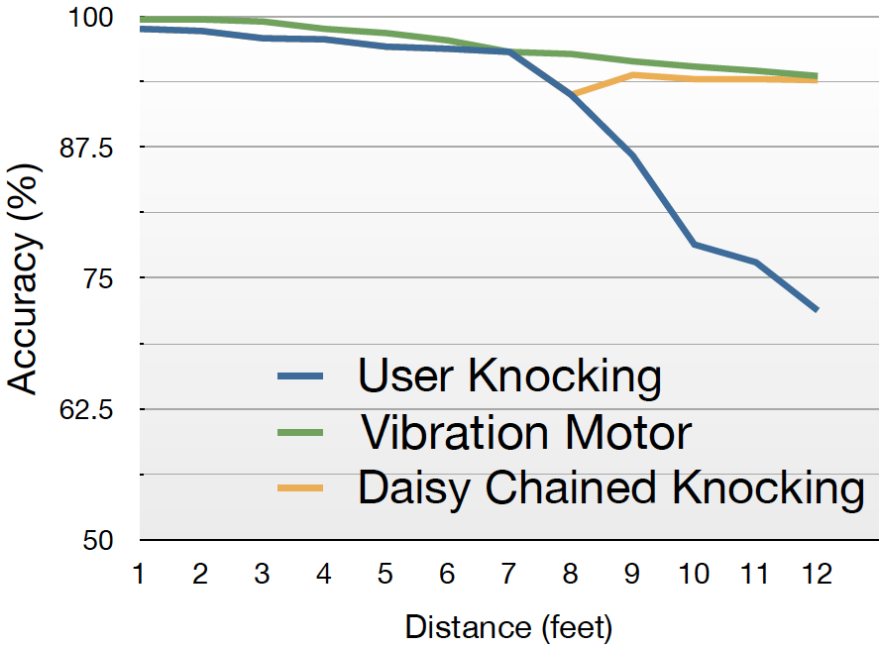

Figure 4.9: Performance of detecting devices on the same surface. User knocking performance drops significantly over larger distance, but this can be ameliorated using daisy-chaining.

smartphones.

4.1.3.1 Detecting Devices on the Same Surface

The device detection system can operate using user-induced or device-induced vibrations. In the evaluation, the order in which these two vibration sources were used was randomized. The experiments were performed on three different surfaces: wood, laminate, and metal. When using user-induced vibrations, the participants were asked to put the phone onto one of the three tables and then they were instructed to knock on the table when the application prompted them. Each table had a host device on it to detect co-presence. When using device-induced vibrations, the devices were simply placed on the table and did not require any user intervention. All the data was recorded in the application. In post processing, I checked if the system was able to classify the user's phone into respective clusters for each of the three tables. Figure 4.9 shows the

performance of SurfaceLink over varying distances between the client and host device. The accuracy of the system expectedly decreases as the inter-device distance increases. When using user-induced vibrations, the accuracy drops significantly around 8 feet, whereas in the device-induced vibration case, the accuracy remains above 94% even at distances more than 12 feet. In order to alleviate the accuracy drop after 8 feet while using user-induced vibrations, I used vibration daisy chaining as described previously. Daisy chaining improves accuracy from 71.9% to 93.9% at 12 feet. Overall, SurfaceLink detects devices on the same surface with 97.7% when the devices are up to 8 feet apart. The performance difference was not significant for any of the materials as the magnitude of vibrations was high in all cases.

4.1.3.2 Detecting Surface Gestures

The data collection application for gesture classification recorded the vibrations observed by the contact microphone. I did not formally evaluate the performance with the on-device microphone because the noise cancellation capabilities of modern mobile devices are part of the hardware firmware and are not accessible to the developer. Four phones were placed on a surface in an arbitrary square with approximately 0.5 m between them.

All possible combinations of the gestures were not evaluated because it would have led to 180 different input gestures and impractically long evaluation sessions. We tested all combinations of *Gesture Shape* and *Touch Modes*. In case of *Gesture Class* and *Gesture Length*, the gestures were only performed with the fingertip. Thereby creating a vocabulary of 5X3+6+3=24 different gestures. Each gesture was repeated 10 times, and the order was randomized. Again, in order to evaluate the performance of SurfaceLink on a variety of surfaces, the data was collected on three different surfaces: wood, laminate, and metal. Figure 4.10 shows the performance of SurfaceLink in classifying different surface gestures using a 5-fold cross validation. The performances of wood and laminate were similar to each other (94.2% and 94.9%, respectively) and were both better than metal (81.2%). This is expected as the metal surface is not a textured surface and was used

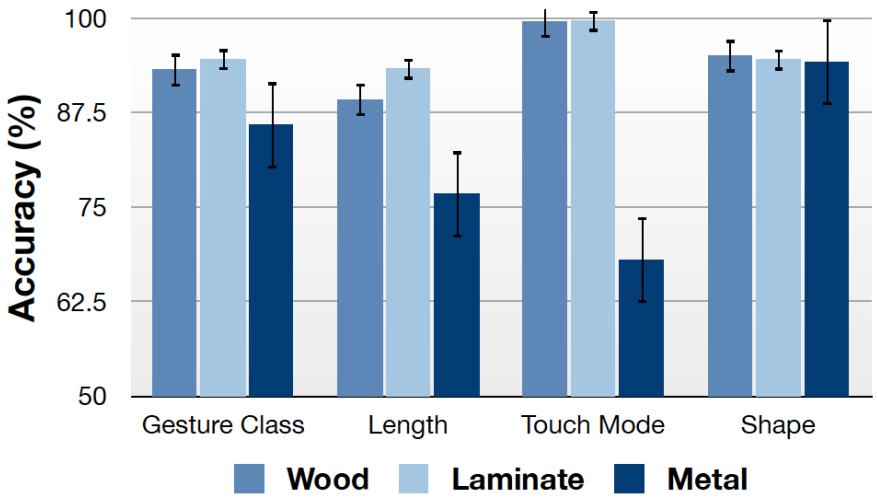

Figure 4.10: Performance of gestures on different surfaces (5-fold cross validation). Wood and laminate performed the best. Errorbars are standard errors.

to only to test the lower bounds of SurfaceLink's performance. The mean accuracy across gesture class, length, touch mode, and shape detection was 90.3%.

The confusion matrices for *Gesture Class* and *Gesture Length* are shown in the Figure 4.11. It can be seen that there is no pair of gesture classes that often confused. The disambiguation of participating and non-participating devices is also very robust. In the evaluation, each user was asked to perform each gesture 10 times, and then we used a 5-fold cross-validation in order to obtain results. In a real world setting the user would need to perform a training phase before using the system. We therefore did an analysis to see how much training data is needed to obtain reasonable performance, as shown in Figure 4.12. The x-axis shows the percentage split in training and test data. Larger values on x-axis signify more training data. It is clear from the figure that in the case of *Gesture Class, Touch Mode*, and *Shape*, SurfaceLink does not require much training data, as just a few examples would suffice. The accuracy is lowest for *Gesture Length*, but even then only 5 examples are needed to increase the accuracy to 85%.

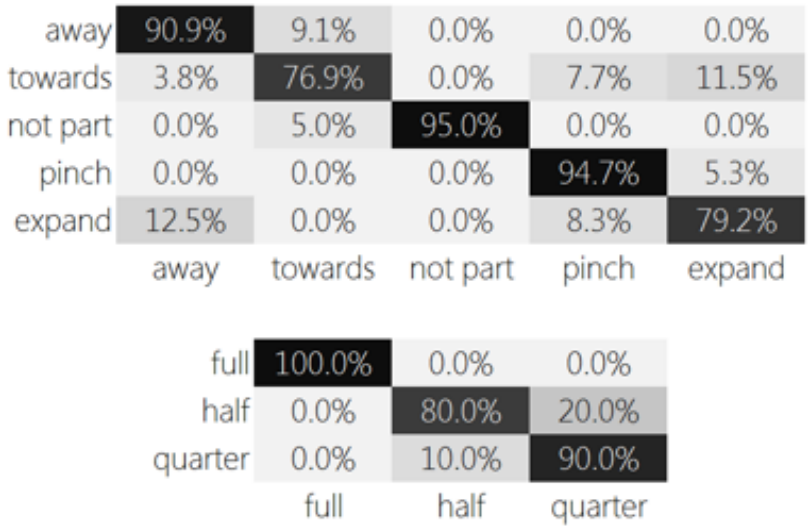

Figure 4.11: Confusion matrices for different (Top) Gesture Class and (Bottom) Gesture Length. (Note: Non-participating devices are abbreviated as not-part)

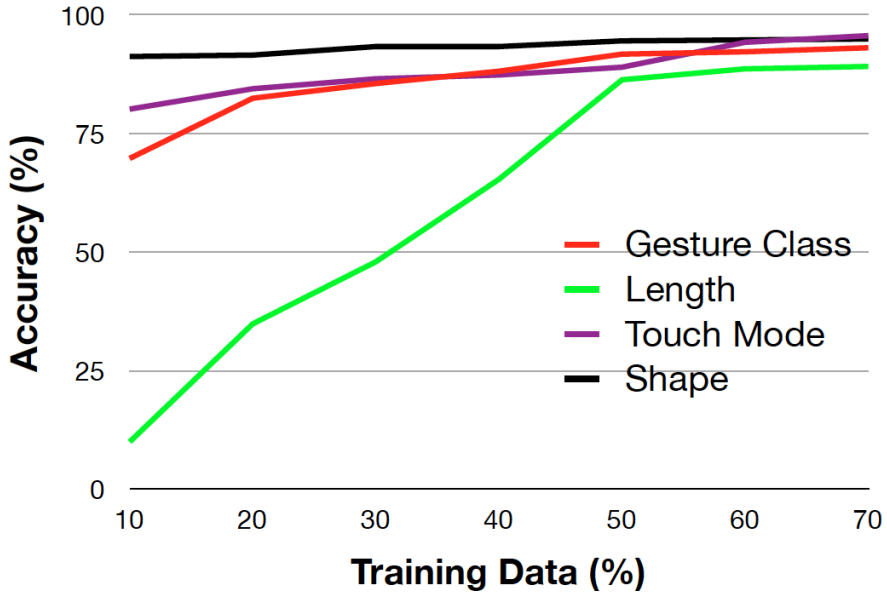

Figure 4.12: Improvement in average accuracy of gesture detection on the three surfaces with increasing size of training data.

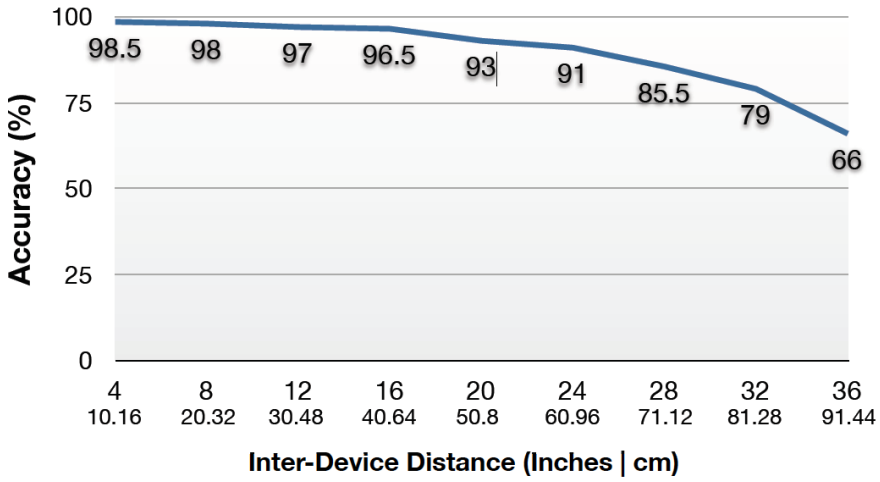

Figure 4.13: Performance of acoustic stereo positioning.

4.1.3.3 Detecting Device Arrangement

This evaluation was divided into two parts. The first part was to evaluate the performance of acoustic stereo positioning. A mobile device was kept at different relative locations to the host device (Motorola Xoom). The aim of this evaluation was to check the bounds of distance between the two devices where SurfaceLink is able to determine effectively determine the arrangement. Figure 4.13 shows the performance of acoustic stereo positioning. The average accuracy over inter-device distances between 4 and 36 inches is 89.4%. The accuracy drops to approximately 66% when the devices are 36 inches apart. This is because the distance between the two speakers on the host device of less than 12 inches. As the inter-device distance increases, the difference in the observed tone amplitude will become negligible. The acoustic stereo positioning detects which devices are to the left and right of each other. But in order to get the accurate arrangement of all the devices on a surface, a combination of stereo positioning and user gesture is required. The participants were asked to draw a gesture between all the devices on the table with their fist. They were presented with 10 unique device arrangements including devices in a straight line, square, triangle, *etc.* The inter-device distance was limited between 4 to 24 inches as larger distances meant devices were too far away for a single user. The participants were not informed which gesture shape to perform. They selected the gesture

shape on their own from a set of triangle, square, circle, line, and arc. They were informed to select the shape keeping in mind that their gesture should try to go near all the devices. For example, when the devices were kept in a triangle arrangement, a circular or triangular gesture would be appropriate.

The system tried to determine the order and whether the devices were in a linear, circular, triangular, or square arrangement. The system accurately detected the device arrangement 88% of the times.

4.1.3.4 Qualitative System Evaluation

I developed a multi-device photo-sharing application (Figure 4.2) in order to qualitatively evaluate the performance of interaction techniques introduced by SurfaceLink. In this application, the system first detects the devices that are present on the same surface. Then it allows the users to interact with devices through surface gestures.

The performance of this system is compared with Bump[1] and Cooperative-Stitching [95]. The purpose of this evaluation was to test how SurfaceLink performs with increasing number of users and devices. I recruited 10 users to evaluate a system in which they transferred pictures between four devices using all the three techniques (*i.e.,* Cooperative-Stitching, Bump, SurfaceLink). The participants were given a randomly generated list of 15 photo transfer tasks. An example of a task will be "Transfer photo from A to B". In 5 of the 15 tasks for each participant, there were multiple receiving devices. For example, "Transfer photo from A to B and C". These tasks were added to mimic a scenario where users might want to share files with multiple devices. In Cooperative-Stitching the users collaboratively drew a line with their finger from sending device to receiving device. For SurfaceLink the users used the swipe gesture to share photos between devices and drew circles with fist to create a subgroup of devices (in the case of multiple receiving devices).

[1]Bump: http://www.bu.mp (Discontinued)

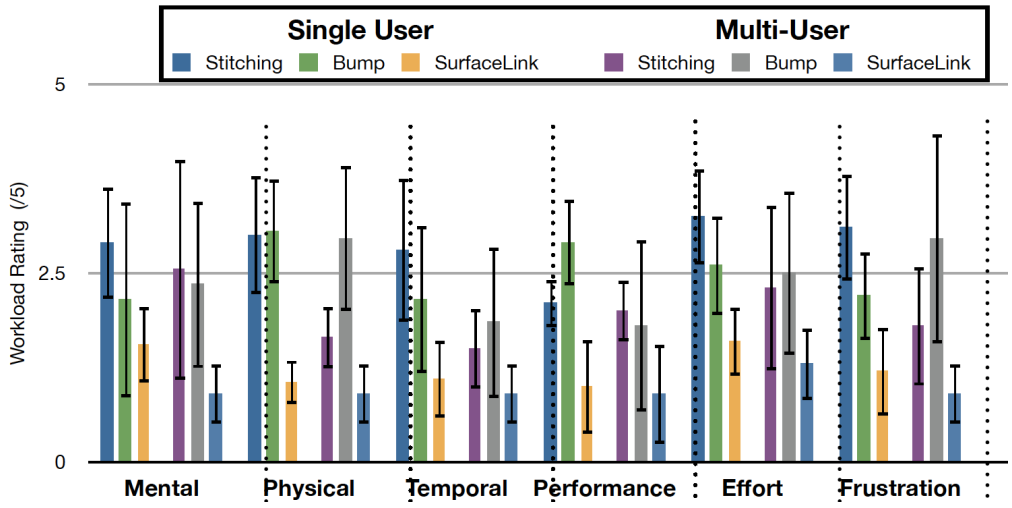

Figure 4.14: Performance of acoustic stereo positioning.

Two experiments were conducted to test the three techniques in single- and multi-user environments. In the first experiment, a single user interacted with 4 devices. In the second experiment, a set of four users controlled one device each. I overlapped the users and each user was part of 2 separate sets. Therefore, there were five sets of four users, each. The participants were taught the functionality of each app and were given time to get familiarized with the interface. Once they were comfortable they were presented with the three techniques in a random order. After completion of tasks with each technique, we asked participants to make Likert-scale ratings (Figure 4.14) based on NASA TLX perceived workload index [88]. I also recorded the time taken by each participant to complete the tasks with each technique.

A repeated measures ANOVA revealed that, regardless of number of devices per user, participants spent significantly less time in completing the tasks while using SurfaceLink as compared to the other two techniques ($F_{2,18} = 10.94$, $p < .05$). A Friedman test showed that in the single user scenario, SurfaceLink resulted in significantly less workload in all dimensions as compared to Bump and Cooperative-Stitching. In the case of multiple users, SurfaceLink resulted in significantly less mental, temporal, and physical workload, and frustration. All Wilcoxon pairwise comparisons were protected against Type I error using a Bonferroni adjustment.

When asked which of the techniques they preferred, and all participants preferred using SurfaceLink over the other two techniques. P4 said, "*The surface gestures feel very natural.*" P2 said, "*The fact that I do not need to interact with all of the devices' touchscreens to share files can be very useful.*" However, 6 out of 10 participants preferred Bump or Cooperative-Stitching over SurfaceLink if they just had to send one single batch of files to a single paired device.

4.1.4 Discussion

The results show that SurfaceLink performs better than Bump and Cooperative-Stitching when there is more than one device per user. There is no established baseline for the interaction techniques for multiple devices. Bump and Cooperative-Stitching were selected as points of comparison because they are very lightweight and do not require any hardware adjustment to the devices. Moreover they are good representatives of "co-occurrence" based multi-device systems. I do not aim to hypothesize that SurfaceLink is better than all the multi-device interaction approaches. I solely wanted to test how SurfaceLink performs with increasing number of users and devices in comparison to some of the proven techniques. The fact that all the participants preferred SurfaceLink suggests that it is an attractive interaction system as the number of devices as well as our interaction with these devices increases.

I deployed the system on a work desk for a week with almost no false positives (4 in whole week). SurfaceLink is able to use the duration of gesture to easily remove false positives. Additionally, considering it is aware of device arrangement and also classifies devices into "non-participating devices" category, accidental touches are almost always classified as "non-participating devices" for all devices.

Extended use of the microphones and accelerometer can have significant power implications. The devices do not need to be in permanent listening mode. The devices can do opportunistic sensing and would only listen when they infer they are on a flat surface.

Surface gestures are also useful in "around-the-device" interactions. Such interactions are becoming very popular. SideSight [30] is one example of such a system. The rich set of gestures introduced by SurfaceLink is demonstrated to be useful in a multi-device interaction scenario. But these gestures can be very useful for a single device interaction as well, and aid in "around-the-device" interaction. I use the photo sharing as an example scenario but the demonstrated gesture language can be used for a variety of purposes in both single- and multi-device environments.

4.2 Going Beyond Devices that Share a Surface

With the proliferation of smart devices (*e.g.,* personal computers, smartphone, smart TVs, other IoT devices) - it is becoming increasingly important for these devices to connect to each other and share information. Although SurfaceLink facilitates a very seamless, context-aware connection between some of these devices, the devices will not always be on the same surface. For example, connecting a smartphone to a TV in order to stream contents on the TV is a common usecase, however a technique like SurfaceLink will not prove very useful here. Numerous techniques have been proposed for selecting objects in the physical environment. However, many solutions require extra sensors to identify devices [183], or they require simultaneous movement of devices [189], which is not always feasible - especially with larger devices such as televisions. There are a number of camera-based solutions but blinking LEDs and QR codes do not necessarily blend into or *disappear* into the environment. In order to solve some of these challenges, I worked with some of my colleagues to build *DopLink*. DopLink uses the mobile device's on-device audio hardware to determine if a particular device is being pointed at by another device. Figure 4.15 shows the gestures supported by DopLink. With DopLink, the user moves their mobile device in the direction of the device they want to connect to. It uses the well-understood phenomenon known as Doppler effect, which characterizes the change in observed frequency of a sound wave as a source moves towards or away from the receiver. DopLink is much less susceptible (when compared to camera-based approaches) to occlusions between the controller and receiver, and is invisible to the user.

Figure 4.15: ***Top*** **Scanning gesture.** ***Bottom*** **Pointing Gesture:** ***(Left)*** **Flick,** ***(Right)*** **Push.**

When the user wants to connect to a partcular device, they hold a button to initiate the inaudible tone (18 kHz) from their phone and then make a pointing gesture towards the target device (Figure 4.15, *Bottom*). A Doppler shift wil be observed on a number of nearby devices, the intended target will receive the maximum frequency shift. All receiving devices continuously sample their microphone at 44.1 kHz. After every 100 ms, all receiving devices report their frequency shifts to the central server (implementation details can be found in [12]). The server compares the frequency shifts and informs the receiver and sender device. Now a connection can be set up between these two devices and a multi-device pairing can be seamlessly accomplished.

The system was evaluated with 6 participants (3 males, 3 females), and an overall accuracy of 95% was observed for selecting a device from a set of 3 target devices. The reliability of the sensing went significantly down if the angle between the two target devices was too small. It was found that the devices need to be at least 20° away from each other for acceptable performance (90% selection accuracy).

Since the development of DopLink, I have worked with some of my colleagues to extend DopLink to enable in-air gestures between all kinds of mobile devices. The resulting

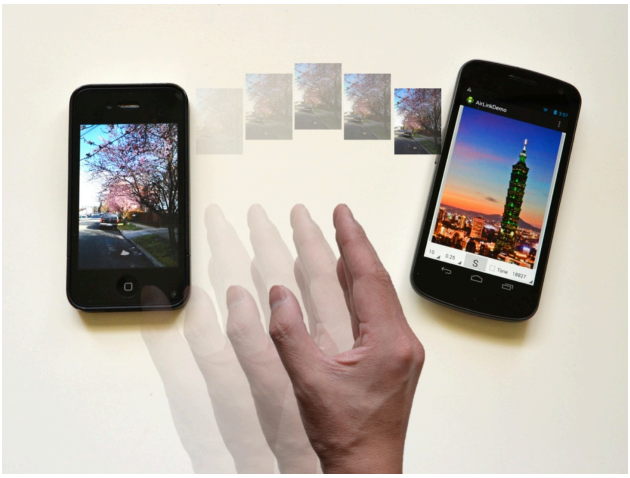

Figure 4.16: AirLink allows file sharing between multiple devices by using in-air gestures through the devices' in-built audio hardware.

system, called *AirLink* [36], uses ultrasonic tones (18-20 kHz) to connect multiple devices and share files between them. By waving the hand from one device to another, users can easily exchange information such as photos between the devices. In AirLink, each device generates an inaudible tone and when the user wants to share a file, they simply wave their hand from the sending to the receiving device (Figure 4.16). The hand movement reflects the ultrasonic waves, causing a frequency shift. This change in frequency is observed on the devices' microphones. By comparing the magnitude, direction, and time of arrival of these frequency shifts, one can identify the direction of the in-air gesture. The implementation details of the work can be found in [36]. AirLink was evaluated in a user study with 11 participants (7 male, 4 female) in a 3-phone scenario. The users performed six different in-air gestures shown in Figure 4.17. Each participant performed 10 repetitions of each gesture and the order of the gestures was randomized. AirLink was able to differentiate between the six gestures with an average accuracy of 96.8%.

Overall, both DopLink and AirLink demonstrate that the Doppler effect based near-ultrasonic audio sensing can be used to improve the multi-device interaction and make it more seamless. However, these are still very early explorations and much more detailed analysis of their performance in noisier and more realistic environments is necessary before they can be used by the end-user on daily basis.

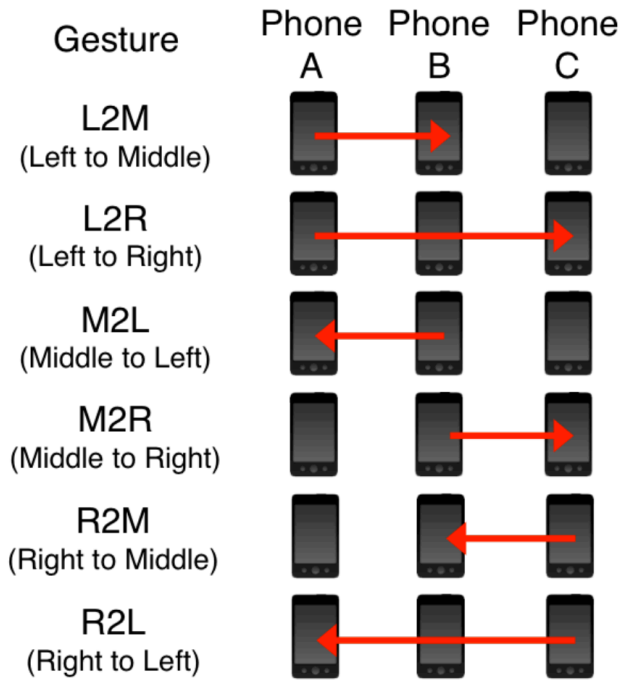

Figure 4.17: By combining 3 basic hand movements: *toward*, *away*, and *toward and away*, multiple inter-device gestures can be formed

4.3 Consequences & Summary

It is common to have multiple computing devices in the same environment. It is also becoming increasingly common for a single user to have multiple devices. Current multi-device interaction technologies do not scale well to increasing numbers of devices per user. In this chapter, I discussed *SurfaceLink*, a system that uses the shared surface between devices as a bounded communication medium. It provides an end-to-end solution for impromptu, scalable interaction between devices using on-device accelerometer, vibration motor, speakers as well as an off-device contact microphone. SurfaceLink detects devices on the same surface (*Acc.* = 97.7%), enables a variety of continuous multi-touch gestures (*Acc.* = 90.3%), and infers relative device arrangement (*Acc.* = 89.4%). The evaluation also demonstrates that SurfaceLink scales well to increasing numbers of devices and users and can be an attractive interaction system as the number of devices or our interaction with these devices increases.

I then briefly discussed *DopLink* and *AirLink*. These efforts use in-air gestures to improve multi-device interaction. This is especially important because there are many co-located devices that do not share the same surface.

Therefore, this chapter demonstrates a very different performance requirement as compared to the previous chapter: In this case, the inference needs to be immediate and very precise. When a user performs a gesture, the system needs to react immediately and the level of precision and accuracy requirements are extremely high. The combination of these two chapters on the applications for Interaction Techniques demonstrate that the on-device sensors can be used as substitutes for dedicated sensors.

These efforts present a great opportunity to improve the way we interact with multiple devices in our environment; especially because these use the sensors that are already on our devices. However, as has been discussed earlier in this thesis, these sensors can be incredibly noisy. These sensors are affected by many little changes in position, or sound, *etc.* It is precisely for this reason why I did the week-long study on a desk to study SurfaceLink's resistance to noise (discussed in Section 4.1.4). SurfaceLink was able to combine the information from the phone's microphone and accelerometer to counter the ambient noise. This highlights one of the dissertation questions "Can multiple on-device sensors be combined to improve performance?".

In many other cases though, countering noise in the signals received through these sensors won't be so straightforward and might involve modification of the current on-device sensors and their infrastructure. These requirements will become even more important when these efforts are deployed and used in a real world setting. The work that I have described in the Part I of this thesis has focused on Interaction techniques, and the work was not deployed for long periods in the real world.

In the next part of this thesis, I will discuss how we can use the on-device sensors for health application. Much of that work is currently deployed in places around the world. There I will go into more detail about using multiple information sources (and sensors) to

counter the noisy estimates from the on-device sensors. I will also base a number of my recommendations for future devices through deployments of these health sensing works.

Chapter 5

Pulmonary Monitoring

Respiratory diseases are among the leading causes of death worldwide. According to a 2008 WHO survey [155], pulmonary ailments (*e.g.,* pneumonia, lung cancer, chronic obstructive pulmonary disorder) account for more than one-sixth of global deaths. Most of the lung ailments are widespread and chronic; therefore the fatality rate can be mitigated by (1) diagnosing the diseases early through effective screening processes, and (2) helping patients manage their ailments in order to prevent them from exacerbating. Spirometry is a most widely employed objective measure of lung function [219] and is central to the diagnosis and management of lung ailments, such as asthma, COPD, and cystic fibrosis. However, high cost and limited availability of spirometers makes spirometry challenging. To this end, in this chapter I discuss a mobile phone-based spirometer that simply uses the on-device microphone to calculate the user's lung function. The smartphone version of this tool is deployed in various parts of the world and apart from making lung quality assessment more widespread, it serves as an example of challenges faced by systems that only use on-device sensors. This chapter also discusses approaches on how some of these challenges can be mitigated through the use of other on-device sensors or user interface modifications, *etc.* (Section 5.5.2).

Figure 5.1: User using SpiroSmart. The maneuver on SpiroSmart is exactly same as on a clinical device, except the lack of mouthpiece.

5.1 Introduction

Lung ailments are one of the leading causes of global deaths. Early stage diagnosis of these diseases can prevent a significant number of complications; however it extremely hard to diagnose these diseases. Chronic obstructive pulmonary diseases, for example, is vastly under-diagnosed and roughly 50% of the patients are unaware of their condition [188]. The challenges stem more from the economical and logistic point of view, rather than technological. The most widely employed objective measure for lung quality is spirometry and is central to the diagnosis and management of a number of lung ailments. Spirometry is usually performed in medical offices and clinics and barring some usability and training issues, are generally regarded as useful for lung quality assessment. However, these devices are usually very costly (upwards of $5,000 USD) and can be quite big. Therefore it is not easy to find a spirometer in resource constrained environments. Moreover, considering the ailments are mostly chronic, the patient needs to monitor and manage their condition frequently and over a long time period. For more frequent tracking, home spirometry with portable devices is slowly gaining acceptance [26, 169]. Measurements at home allow patients and physicians to more regularly monitor trends and detect changes in lung function that may need evaluation

and/or treatment. Home spirometry has the potential to result in earlier treatment of exacerbations, more rapid recovery, reduced health care costs, and improved outcomes [115, 155, 201, 203]. However home spirometry currently faces challenges such as cost, patient compliance, usability, and an integrated method for uploading results to physicians [66, 66]. Moreover, when done at home, patients have no coaching or feedback or quality control mechanisms.

In order to solve some of these challenges around spirometry, in this chapter I will discuss *SpiroSmart* (Figure 5.1), a smartphone-based approach that measures lung function using the phone's built-in microphone (*i.e.,* a complete software-enabled solution). The spirometry maneuver on SpiroSmart is exactly the same as on a conventional, clinical device. SpiroSmart requires the user to hold the smartphone at approximately arm's length, breathe in their full lung volume, and forcefully exhale at the screen of the phone until the entire lung volume is expelled. The phone's microphone records the exhalation and sends the audio data to a server, which calculated the exhaled flow rate by estimating models of the user's vocal tract and the reverberation of the sound around the user's head. SpiroSmart is able to compute and provide flow rates and graphs similar to those found in home or clinical spirometers (Figure 5.2). In our controlled trials, SpiroSmart's performance has been very similar to the clinical devices.

There are distinct advantages to developing a smartphone-based solution as compared to current commercial spirometers. Firstly, the low-cost and inherent portability of the mobile device allows for a much greater uptake and accessibility. The relative low cost as compared to spirometers can also help in lowering access barriers to spirometry in resource constrained environments. Secondly, a smartphone spirometer can have built-in coaching and feedback - mechanisms to maximize measurement acceptability that are critically lacking in current spirometers. Finally, with the smartphone, spirometry can be easily coupled with evaluations such as symptom scores and diaries, cough sensing, or oximetry to provide a comprehensive disease self-management tool. Currently SpiroSmart is deployed in various locations in the Seattle area in USA, and in India and Bangladesh. To date, it has been used by over 5,000 patients.The aim of these

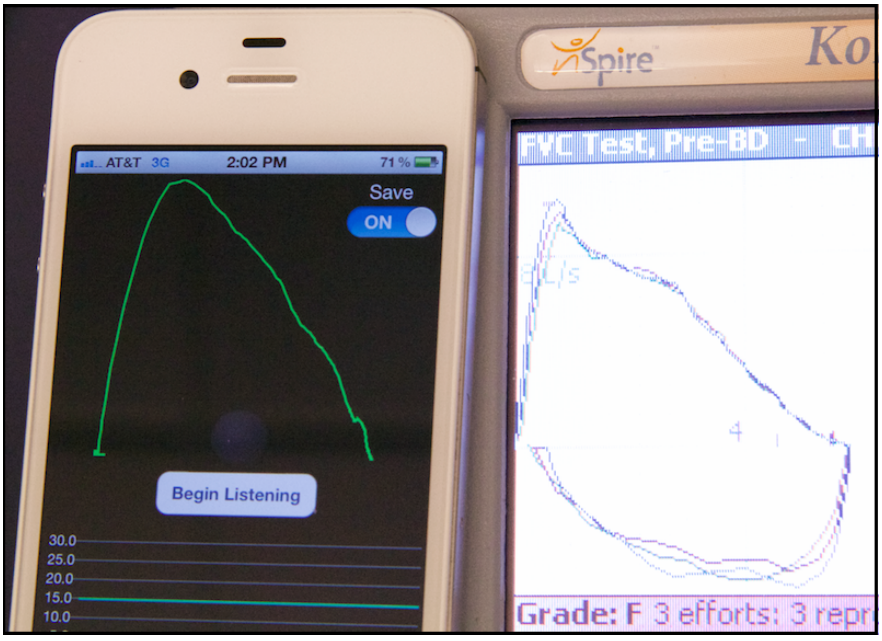

Figure 5.2: Example output from *(Left)* SpiroSmart and *(Right)* a clinical spirometer.

deployments is to compare the performance of SpiroSmart and clinical spirometers. While the analysis of these deployments is not part of this thesis, in Section 5.5 I will discuss the opportunities, challenges, and lesson learned from these deployments.

One of the biggest challenges for SpiroSmart is that it requires a smartphone. Smartphones are becoming prevalent at a breathtaking rate, yet more than half of the mobile phone users in sub-Saharan Africa and South Asia will still be using a non-smartphone (or feature phone) in 2020 [62]. A major portion of the population suffering from lung impairments lives in these low resource environments. In fact, according to a recent WHO report [238] more than 90% COPD deaths occur in low and middle-income countries. Thus, there is a significant case for a technology like SpiroSmart to be more inclusive and it needs to work on all mobile phone, and not just programmable smartphones. To this end, in Section 5.6 I will discuss *SpiroCall*, a call-in server that measures lung function on *any* phone without the need for a locally running application. Unlike SpiroSmart, it transmits the collected audio to the server using the standard voice telephony channel. Once the server computes the results, it reports them

back to the user via audio or text message. Although the call-in service removes the need for a smartphone, there are other significant usability challenges that are more difficult to mitigate: how a user holds the phone (angle, microphone occlusion, *etc.*), the distance from the user's mouth to the phone, and how wide a user opens their mouth. While controlling some of these issues is possible with a smartphone and its plethora of on-device sensors, they are difficult to mitigate on a feature phone. Recognizing that some people may not be able to master the technique needed to perform the spirometry maneuver, I also designed a simple and low-cost 3D-printed whistle accessory. The whistle generates vortices as the user exhales through it, changes its resonating pitch in proportion to the flow rate. The whistle does not have any moving parts and it is as simple as any spirometer mouthpiece. SpiroCall and the whistle accessory were evaluated in a controlled study with 50 participants and the results were very similar to SpiroSmart *and* conventional spirometers. The whistle performed more consistently for people with lower lung function and produced fewer over-estimations of lung function (*i.e.,* false negatives), as compared to when not using a whistle.

Overall, the combination of SpiroSmart and SpiroCall promises to lower the access barrier to spirometry. Although significant usability challenges need to be addressed, eventually these efforts might help in solving some of the very hard problems associated diagnosis and management of chronic lung ailments.

5.2 Pulmonary Ailments and Spirometry

Chronic lung diseases are the third most common cause of death in the world [1,60,125]. In fact, COPD is already the fourth leading cause of death and is rapidly overtaking infectious lung diseases, such as pneumonia and influenza [161]. Most of these lung diseases are chronic and cannot be cured. The patients need to be diagnosed quickly and then they need to be helped to manage their condition for the rest of the patient's life. The problem is twofold here, (1) the diseases need to be diagnosed reliably and quickly, and (2) monitoring and managing the condition properly [242].

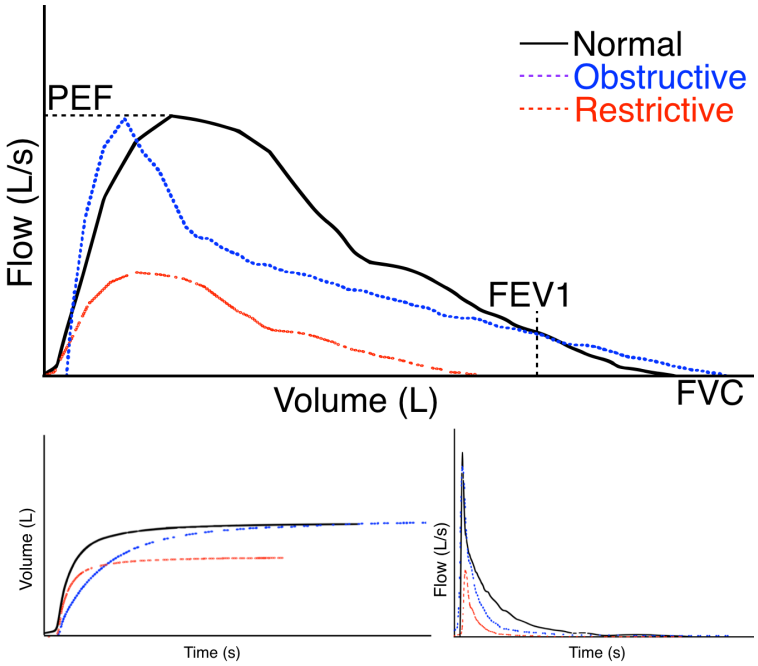

Figure 5.3: ***(Top)*** **Examples of different Flow *vs.* Volume curves and major lung function measures.** ***(Bottom)*** **Volume *vs.* Time and Flow *vs.* Time curves**

Most common chronic lung diseases can be classified as either *obstructive* or *restrictive*. In obstructive diseases, exhaled air comes out more slowly than normal. The air flow is impeded between the bronchi and bronchioles through collapsed or inflamed airways. People with obstructive lung diseases usually have shortness of breath, coughing, and wheezing. On the other hand, restrictive lung diseases are usually due to stiffness in lungs or reduced lung expansion. In these cases, the patients are unable to inhale as much air as someone with healthy lungs. Stiffness of the chest wall, damaged nerves, or weak muscles can also cause restriction in lung expansion. Cystic fibrosis is an example of restrictive disease and is characterized by increased amount of mucus in lungs. For both obstructive and restrictive disease, the earliest noticeable symptom for the patients is shortness of breath and chronic coughing (more prevalent in restrictive diseases).

The standard of care for diagnosis and management of chronic lung diseases is pulmonary function testing (PFT) or spirometry [155, 242]. In a pulmonary function test, the patient inhales as much air as they feel they can, and then they forcefully exhale

through a mouthpiece. The patient keeps exhaling until they do not feel any air left in their lungs. As the performs the maneuver, the machine records the flow rate and generate a progression and flow and volume over the duration of the test. A standard spirometer measures flow rate of air as it passed through a mouthpiece. This flow can be integrated from the mouthpiece to achieve Flow *vs.* Time (FT), Volume *vs.* Time (VT), or Flow *vs.* Volume (FV) plots of the expiration. Figure 5.3 shows examples of these three plots. In a healthy individual the rise to peak flow rate (PEF) is very steep, and the decay from PEF is almost exponential in time. The FV curve plots the exponential decay in flow rate against its integral (volume) and represents the linearization of exponential expiratory decay. Hence, for a healthy individual the descending limb of the FV plot is almost a straight line (black, solid line in Figure 5.3, *Top*). As obstruction to the airflow increases, the flow rate decreases faster than exponentially (Figure 5.3, *Bottom-Right*) after reaching its maximum value (PEF). Therefore, it attains a curved or "scooped" slope (blue, dashed line in Figure 5.3, *Top*). For an individual suffering from a restrictive lung disease, such as cystic fibrosis, the respiratory muscles weaken and the patient's lung capacity (FVC) decreases (red, dashed line in Figure 5.3. *Top*).

Apart from generating these plots, spirometers also produce a number of lung function measures. Four of the most important lung function measures are:

1. **Forced Vital Capacity (FVC):** The total volume of air expelled during the expiration,
2. **Forced Expiratory Volume in one second (FEV_1):** The volume of air expelled in the first one second of the expiration,
3. **FEV_1 /FVC:** Ratio of FEV_1 and FVC, and
4. **Peak Expiratory Flow (PEF):** Maximum expiratory flow rate reached during the test.

The first three measures, FVC, FEV_1 , and FEV_1 /FVC are considered essential spirometry parameters. They are used to quantify the degree of airflow limitation due to reduced airway diameter. Generally a healthy individual's lung function measures are at least 80% of the values predicted based on their age, height, and gender [119]. For example,

obstructive diseases affect FEV_1 and FEV_1 %, while PEF and FVC remain unaffected. The FEV_1 % is monitored closely in these situations. Abnormal values are (expressed in percent of predicted value based on height, age, and gender (and sometimes race)) [155]:

- Mild Lung Dysfunction: 60-79%,
- Moderate Lung Dysfunction: 40-59%,
- Severe Lung Dysfunction: below 40%.

5.2.1 Modern Spirometry Devices

Modern spirometers are generally flow based, measuring the instantaneous exhaled flow (liters/sec.). There are four prevalent types of flow-based spirometers: pneumotachographs, turbines, anemometers, and ultrasounds. Pneumotachs measure the pressure differential across a membrane as the subject exhales. These devices are affected by humidity and temperature and require daily calibration. Pneumotachs are the most prevalent spirometers in medical offices and clinics because of their accuracy. High-end clinical spirometers can cost upwards of $5000 USD and be comparable in size to a small refrigerator. The patient sits inside an enclosure that controls humidity, temperature, and oxygen levels. Portable, ATS-endorsed spirometers (about the size of a laptop) generally cost between $1,000-$4,000 USD, and although they are relatively portable compared to their counterparts, they are still bulky, complicated devices.

Low cost peak flow meters exist, which can only measure PEF ($10-$50 USD). They are about the size of a baseball and typically use a mechanical apparatus without any electronics. PEF in isolation, however, is generally considered to be a poor indicator of lung function [174]. Digital home spirometers that report only FEV_1 are also commercially available ($50-$200 USD). These meters vary widely in their reporting and archiving of results—some require patients to manually write down the values in journals or have a USB desktop connection. The newest models (*e.g.,* SpiroTube) can connect to a mobile phone or laptop via Bluetooth, but cost considerably more, from $900-$3500 USD.

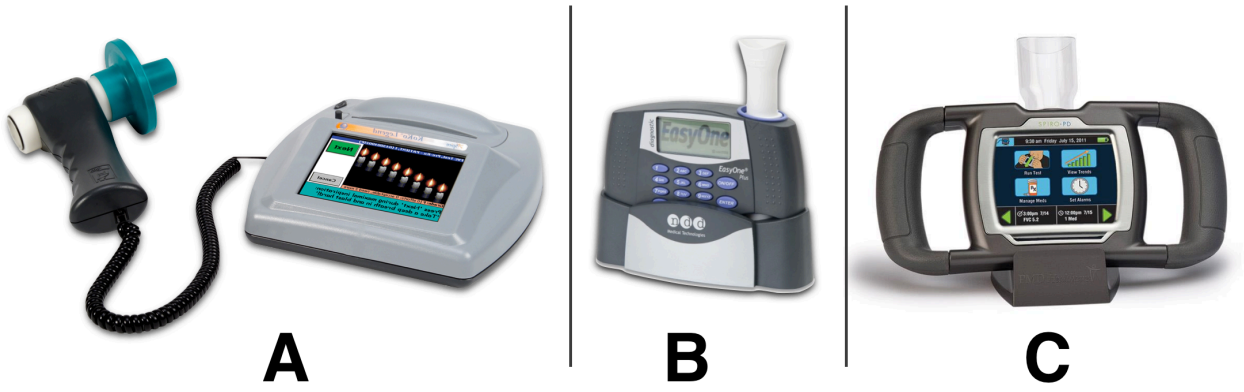

Figure 5.4: Three FDA-cleared spirometers: *(A)* KoKo Legend, *(B)* nDD EasyOne, *(C)* Spiro PD

For comparison with SpiroSmart, I have used three portable spirometers in different trials (Figure 5.4). These spirometers are cleared by FDA for clinical use. KoKo Legend[1] (Figure 5.4.A) is the most expensive one in this list and sells for around $1,500 USD. It is pneumotach based and measures the pressure differential across a membrane. In comparison, ndd EasyOne[2] (Figure 5.4.B) retails for around $900 USD and uses ultrasonic transducers positioned on either side of the tube that the patients exhale through. Patient's exhalation disrupts transmitting transducer's generated acoustic signal before it is recieved by the receiving transducer. Analysis of the received waveform yields flow rate with reasonable accuracy. SpiroPD[3] (Figure 5.4.C) is the simplest and least expensive of the three. It is geared towards home use and has the one of the most descriptive user interfaces. It is supposed to be used by a single user and is designed for home use.

MobiSpiro [194] directs the air past a hot film anemometer, cooling the hot film and allowing for accurate estimate of flow rate. While the approach is generally accurate, the cost is dictated by the cost of the sensors, manufacturing, and quality control. Also, anemometers need to be calibrated quite frequently; limiting their usability in home

[1]http://www.nspirehealth.com/products/koko-testing-devices/

[2]http://www.nddmed.com/

[3]https://www.mypmd.com/

environment.

In 2011, Gupta *et al.* presented mobileSpiro, a $100-$200 pneumotach that connects to an Android smart phone [81]. Carspecken *et al.* [32] created a similar device caleed TeleSpiro. These devices measure pressure drop across a tube of known dimensions. These devices are usually connected to the computer or phone over a USB or wireless connection. The cost of manufacturing such devices can be quite low (around $20 USD). However, the user still needs to connect an extra hardware to their mobile devices and that can lower their usability and long term patient compliance. A number of smartphone applications (*e.g.,* mySpirometer, Spirometer Pro) have recently appeared on the iPhone and Android platforms that claim to measure aspects of lung function. However, these applications are advertised as games and have disclaimers warning not to use them for medical assessment.

5.3 Related Work

The growing ubiquity, sensing capabilities, and computation power is enabling smartphones to become a dependable health sensing platform. SpiroSmart draws motivation from prior research on exploring ways to leverage mobile devices for low-cost healthcare solutions.

A number of researchers have evaluated how multiple sensors could be connected to a smartphone via an external board to collect physiological information [27, 195, 245]. Apart from using the on-device sensors for measuring steps, smartphones can also measure other physiological signs such as heart rate. Poh *et al.* [175, 176] developed an electro-optic sensing system that provides Photoplethysmography (PPG) through the user's earlobe. Anderson *et al.* connected several different types of USB ultrasonic probes to a mobile phone for portable, low-cost obstetrical ultrasound imaging.

Many phone-based medical devices leverage the on-device sensors of the mobile devices and do not require external hardware. Poh *et al.* [176] use a camera to detect a person's

heart rate at a distance, and with no contact between the user and the device. Bishara *et al.* [20] modified existing mobile phone camera to perform lens-free holographic microscopy. Bouris *et al.* [22] developed a method to detect retinal cancer using the phone's camera. Smartphones have also been used to improve point-of-care diagnostics by automating analysis of blood and urine samples [206].

5.3.1 Audio-based Health Sensing

Similar in function to SpiroSmart, several technologies use microphones to sense markers of medical health. These are more typically calibrated microphones or microphones with special attachment points. Using in-ear microphones, researchers have detected eating habits (*i.e.,* when and many times what a patient is eating [9, 167]. Yatani *et al.* extended this work to include outer ear microphones and mechanical amplification via a stethoscope [247]. Wheeze detection has also been investigated for diagnosing asthma severity and COPD [100]. Snoring and crackle (high frequency impulses while breathing) have been sensed using bedside [173] and chest worn microphone arrays—which can also pinpoint the area from which the crackle originates. These measures have shown to be clinically useful for diagnosing forms of sleep apnea and possibly monitoring crib death [52]. Respiratory rate is another typical vital sign sensed using bedside and body worn microphones [123]. Respiration, however, is typically much quieter than ambient sounds. As a result, many sensing solutions use an array of sensors like cameras and accelerometers in conjunction with the microphone to measure respiration more reliably [123, 133].

A number of systems exist that measure vital signs and symptoms using low cost microphones (*i.e.,* microphones similar to those found on a phone). For example, ambient audio has been used to detect the mood of patients [223]. Low cost microphones can also be used to detect Korotkoff sounds when checking blood pressure with a cuff [7], although these systems typically work better with amplification via stethoscope. In 2011, Larson *et al.* presented a solution that uses the microphone on the mobile phone to detect and count coughs [131].

5.3.2 Phone and Audio-based Spirometry

In 2012, Brimer *et al.* developed a low-cost spirometer using an acrylic resonance chamber and microphones. Conceptually, their resonance chamber behaves similarly to our design as both whistles change their pitch with change in flow rate. The main difference is that Brimer *et al.* 's design needs the microphone to be directly connected to the resonance chamber and thereby making it unsuitable for using on-device microphones. Xu *et al.* [241] have recently modified a harmonica to enable lung function estimation on a phone. The harmonica changes its amplitude in proportion to flow rate. While this is a very interesting technique for flow estimation, it still relies on amplitude modulation. Wello Labs has also announced an attachment for smartphones to perform spirometry. However, right now it is not clear how it works, as the product is not available.

Vonnegut [224] proposed a vortex whistle design in 1954. SpiroCall uses two larger versions of that whistle, as the maximum flow-rate achieved during a spirometry test would overwhelm the original dimensions. Watanabe and Sato [229] proposed to use a similar design for spirometry, but the design was not rigorously evaluated to understand its medical relevance. Moreover, experiments showed that the suggested dimensions got overwhelmed in many spirometry tests during the pilot study. Finally I designed two different sizes of the vortex whistle for the study and found them to be suitable for spirometry for all 50 participants in the evaluation.

5.4 Initial Prototype

The initial prototype of SpiroSmart was developed in 2012 and was evaluated in a controlled study with 52 participants. It used the phone's built-in microphone and required the user to hold the smartphone at approximately arm's length, breathe in their full lung volume, and forcefully exhale at the screen of the phone until the entire lung volume is expelled. The phone's microphone recorded the exhalation and sent the audio data to the server, which calculated the exhaled flow rate by estimating models of the user's vocal tract and the

reverberation of sound around the user's head. Flow rate was estimated by calculating the envelope of the sound in the time domain; performing resonance tracking in the frequency domain; while measuring white noise gain through linear prediction.

I closely collaborated with experienced pulmonologists from two different hospitals to help inform the design of this initial prototype and to critically compare the accuracy of SpiroSmart to a clinical spirometer (endorsed by the American Thoracic Society or ATS). In a study that included 52 subjects, SpiroSmart had a mean error of 5.1% for the most common measures of lung function. The study showed that SpiroSmart can be used directly out-of-the-box, without any user-specific training or calibration. However, the results indicates that the performance of SpiroSmart improved if we calibrate this generic system for a particular user, decreasing the mean error in estimation of lung function to 4.6%. Lastly, five pulmonologists were asked to make diagnosis using measures and graphs generated from SpiroSmart and from a clinical spirometer. The initial prototype and its evaluated showed that SpiroSmart can be an effective tool for diagnosing not only abnormal lung function but also the degree of obstruction.

5.4.1 Data Collection

To evaluate and inform the design of SpiroSmart, a dataset of audio samples was created. In all, 52 volunteers participated in a 45-minute study session (Table 5.1). All participants self-identified themselves as having none or only mild lung conditions. The custom data collection application for the Apple iPhone 4S recorded subjects' exhalation sounds using the built-in microphone (at 32 kHz) and provided feedback to the user, coaching them through the spirometry maneuver. In the same session, the participants also used an ATS certified standard clinical spirometer, the nSpire KoKo Legend, as the "gold standard". The KoKo is a pneumotach spirometer and was calibrated with a 3 L syringe before each session.

Spirometry measurements are completely effort-dependent and patients are coached through this maneuver by a trained technician. While using the clinical spirometer, participants were coached both orally and with gestures. With SpiroSmart, participants

Table 5.1: Demographics information of the participants

Subject Demographics (N = 52)	
Males (n, %)	32 (61.5%)
Age (yrs) (mean, range)	32 (18 - 63)
Height (cm) (mean, range)	172 (152 - 196)
Reported Lung Ailments (n, %)	
Mild Asthma, 9 (17.3%)	Chronic Bronchitis, 2 (3.8%)
Cystic Fibrosis, 1 (1.9%)	Collapsed Lungs, 1 (1.9%)
Abnormal Curves (n, %)	12 (23.1%)
Wheeze Present (n, %)	25 (50%)
Never Performed Spirometry (n, %)	29 (55.8%)

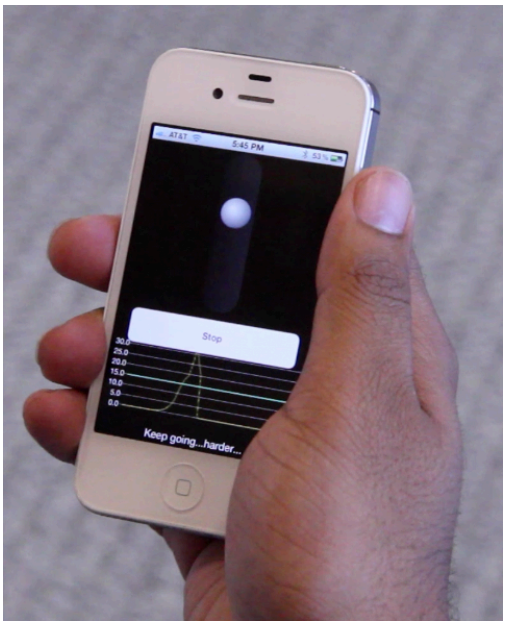

Figure 5.5: SpiroSmart application user interface

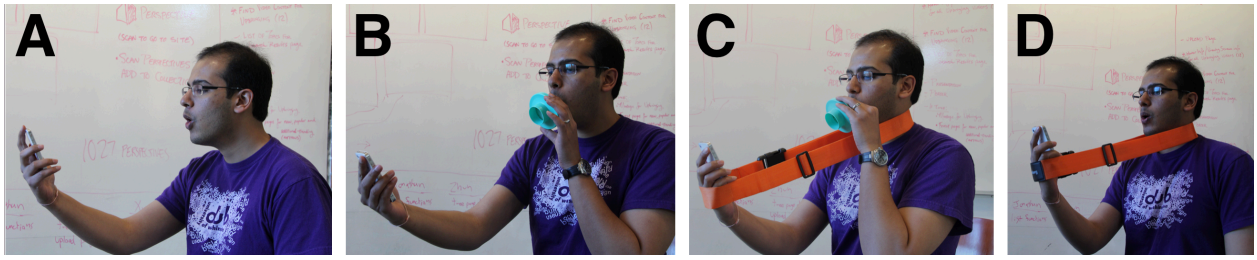

Figure 5.6: Configuration used to test SpiroSmart

were coached with textual prompts on the screen and only with gestures—oral prompts would have interfered with the audio recording (Figure 5.5).

SpiroSmart also calculated a real-time estimate of flow (using LPC gain, discussed in the next section), and displayed the measure as a real-time visualization. This also provided an incentive graphic; namely, a ball displaced vertically in a cylinder proportionally to the strength of the exhalations. After the initial burst, the ball dropped slowly to the bottom of the cylinder, signifying the end of the test. Like the KoKo Legend Spirometer, SpiroSmart displayed an estimated Flow *vs.* Volume curve at the end of the effort (Figure 5.5). An estimate of exhaled volume was calculated by integrating estimated flow with respect to time.

The forced expiratory maneuver was explained to participants and they were asked to practice using the spirometer. Once the participants were able to perform an acceptable maneuver according to ATS criteria for reproducibility, three efforts were recorded using the spirometer [155]. The raw flow and volume measurements from the KoKo were obtained using a USB connection and custom software. Next, participants were introduced to SpiroSmart.

In the pilot study, some participants unintentionally varied the distance at which they held the phone as well as lip posture, potentially introducing unwanted variability. Therefore, in the formal controlled evaluation participants used SpiroSmart in four configurations, in random order: with a mouthpiece (to maintain lip posture), with a sling (to maintain distance), with neither attachment; and with both attachments

(Figure 5.6). Note that it was impossible to collect data from SpiroSmart and the KoKo Legend at the same time so explicit ground truth is unknown. Instead, each effort from SpiroSmart was associated with one randomly selected acceptable curve from the KoKo device during that same session. The signals were aligned using PEF for the KoKo and the maximum amplitude in the audio stream from SpiroSmart as reference points. The audio stream was segmented automatically starting one second before and ending six seconds after the maximum audio amplitude.

10 participants were asked to return for two more data collection sessions (2 days up to two weeks apart). These repeated trials by the same used allowed to evaluate the consistency of measurements from SpiroSmart over longer periods. The participants were asked back based on specific demographics—an equal number of men and women, and equal number of normal and abnormal subjects. In this study, "abnormal" subjects are defined as those with abnormally shaped curves, not necessarily reduced lung function measures. In total, data was collected from 248 clinical spirometer uses and 864 SpiroSmart uses.

Interestingly, 6 subjects were found to have abnormally shaped curves from ailments that they were unaware of and 8 of the 13 subjects who reported lung ailments produced normally shaped curves—albeit with less than expected lung function measures.

5.4.2 Algorithm

The data collection resulted in a dataset of digitized audio samples from a smartphone. These audio samples are uncalibrated, AC-coupled measures of pressure, p(t). These measures need to be converted into measures of airflow at the lips, ulips(t). The main goals, then, are (1) to compensate for pressure losses as the sound travels from mouth to microphone, (2) convert the pressure values to an approximation of flow, and (3) remove the effects of AC-coupling. Pressure losses can be approximated using an inverse model of the sound reverberation around the head. Turbulent airflow, as it passes through a fixed opening (*i.e.,* the mouth), has a characteristic pressure drop that can be used for converting pressure into flow. Lastly, the effects of AC-coupling are removed using signal

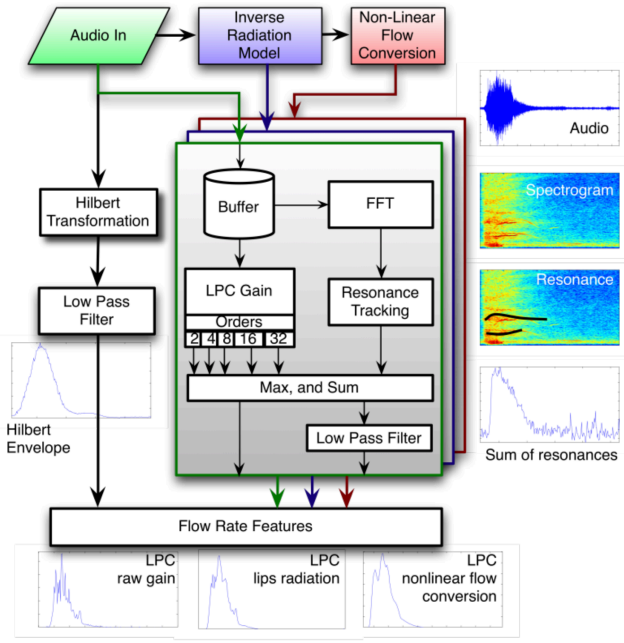

Figure 5.7: Block diagram of SpiroSmart's feature extraction.

power, frequency characteristics, and models of the vocal tract.

Finally, these approximations are combined and the non-linearity is removed using multiple regression algorithms. The methodology is broken into two block diagrams: compensation and feature extraction (Figure 5.7), and machine learning regression (Figure 5.8).

5.4.2.1 Distance and Flow Compensation

The first stage in the processing pipeline (Figure 4) is inverse radiation modeling, which compensates for pressure losses sustained over the distance from mouth to microphone, and reverberation/reflections caused in and around the subject's body. The transfer function from the microphone to the mouth is approximated by a spherical baffle in an infinite plane and is given by [69]:

$$H(e^{jw}) = \frac{P(e^{jw})}{P_{lips}(e^{jw})} \sim \frac{jwC_{head}}{D_{arm}} exp(-\frac{jwD_{arm}}{c})$$

where D^{arm} is the arm length, C_{head} is the head circumference (both approximated from the patient's height using body proportion equations), and c is the speed of sound. The transfer function inverse is applied by converting it to the time domain, $h_{inv}(t)$, and using FIT filtering with the incoming audio. Once applied, the output is an approximation of the pressure at the lips, $p_{lips}(t)$.

This pressure value is then converted into flow rate. For turbulent airflow, the non-linear equation converting pressure drop across the lips to flow rate through the lips is given by [69]:

$$u_{lips}(t) \sim 2\pi r_{lips}^2 \sqrt{2.p_{lips}(t)}$$

where r_{lips} is the radius of the mouth opening (a constant resistance across frequency). Note that some scaling constants from the equations have been removed and the equations are only proportional. Therefore, these estimates can only be used as features and once enough data is collected, appropriate regression algorithms can decide how the features should be scaled and which features are most stable.

5.4.2.2 Feature Extraction

At this point, each measure: $p(t)$, $p_{lips}(t)$, and $u_{lips}(t)$, is a high frequency, AC-coupled signal (Figure 5.7). These signals need to be converted into approximate volumetric flow rates. SpiroSmart uses three transformations to convert the AC-coupled signal into a more stable approximation: (1) envelope detection, (2) spectrogram processing, and (3) linear predictive coding (LPC). The envelope of the signal can be assumed to be a reasonable approximation of the flow rate because it is a measure of the overall signal power at low frequency. Linear prediction can also be used as a flow approximation. It assumes that a signal can be divided into source and a shaping filter. It then estimates the source power and the shaping filter coefficients. The "source variance" in case where the user exhales at the screen is the source of white noise, *i.e.,* the power of generator of white noise or the power of flow rate of the lungs. Finally, in frequency domain, resonances that are excited by the vocal chords can be seem in the spectrograms. These should be

proportional to the flow rates that cause them. The implementation of each of the three transformations is explained next.

5.4.2.2.1 Envelope Detection

Hilbert transform is used to calculate the envelope of the AC-coupled signal. An example example is shown in Figure 5.7. Multiple sizes of the Hilbert transforms and low-pass filters are used and the final estimates are all resampled to maintain same sampling rate as the spectrogram and linear prediction models.

5.4.2.2.2 Spectrogram Processing

During the forced exhalation, the audio from the phone is buffered into 30 ms frames (with 50% overlap between frames). Most tests in the data collection lated from 4 to 7 seconds, resulting in 250-500 frames per exhalation. Each frame is then windowed using a hamming window and the $|\text{FFT}|_{\text{dB}}$ is taken to produce the magnitude spectrogram of the signal. The resonance are then extracted using local maxima in each FFT frame, calculated over a sliding window (callout in Figure 5.7). Any maxima that is greater than 20% of the global maximum is saved. After all frames have been processed, in order to preserve only karge and relatively long resonances, any resonance less than 300 ms is discarded as noise. Finally, the average resonance magnitude in each frame is calculated and saved as a flow-rate estimate (callout in Figure 5.7).

5.4.2.2.3 Linear Prediction Processing

The audio signal is again windowed into 30 ms overlapping frames. For each frame a number of LPC models are taken with filter orders of 2, 4, 8, 16 and 32 (increasing vocal tract complexity). The approximated "source power" that excites the filter is saved for each frame as an approximation of the flow rate. Examples of the LPC from using $p(t)$, $p_{lips}(t)$, and $u_{lips}(t)$ are shown at the bottom of Figure 5.7.

5.4.2.2.4 Post Feature Processing

Once the approximated flow rates are returned, they are denoised using a Savitsky-Golay polynomial filter of order 3 and size 11. This operation fits a 3rd order polynomial inside a moving window and is robust to many types of noise while keeping the relative shape of the most prominent signal intact. The filtered and non-filtered signal are both fed as features to the subsequent regression stage.

5.4.2.3 Machine Learning Regression

The feature extraction results in a number of uncalibrated approximations of the flow rate. These approximations are used as features for two different regressions (Figure 5.8): one to generate specific lung function measures and a second to generate the relative shape of the spirometry curve (flow *vs.* time curve is generated and others are calculated from FT curve).

5.4.2.3.1 Folding

The participants in the dataset are folded into several training subsets, providing a number of diverse models that can be combined to create a global model. For example, one subset randomly divides the participants into ten folds equally. Another subset divides participants with wheezes together into ten folds. Another subset divides the dataset into ten folds, but ensures there are equal numbers of abnormal and normal curves to train on. Each subset is used to create a different regression model and the ensemble can be clustered together to form one decision. Note that for any subset a participant in a testing fold is never used in the training fold. Moreover, to investigate "personalizing" the models, SpiroSmart also creates augmented folds that contain data from repeat sessions (for the 10 subjects who performed three sessions spanning multiple days). In this way, "personalized" models are trained using data from the same participant (but on different days) mixed with data from the general model. The generalized and personalized models are evaluated separately.

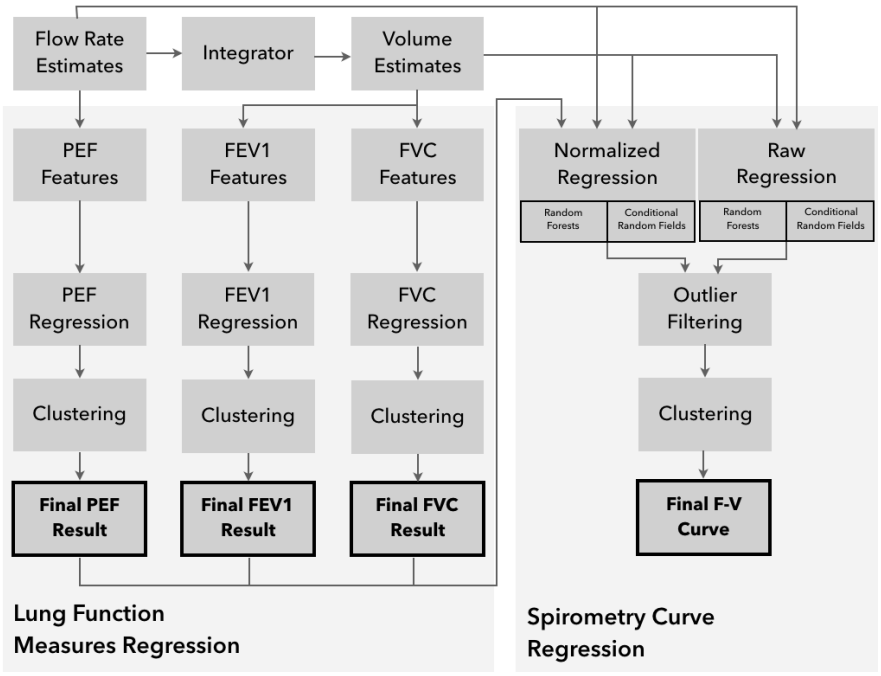

Figure 5.8: Bloack diagram of regression algorithms used in Spirosmart.

5.4.2.3.2 Lung Function Measures Regression

The output of the feature extraction creates a number of features at 15 ms time steps that approximate flow rate over time. If each feature is treated as a flow rate, one can regress directly to PEF, FEV_1 , and FVC. For example, PEF is defined as the maximum flow reached in a single effort. Thus, for a curve in a given fold, if one takes the max of each feature and it is possible to regress to the PEF. Note that integrating flow with respect to time gives volume of air exhaled. Thus for FVC, the integration of each feature is taken. For FEV_1 , the integration of the features during the first second is used (Figure 5.8).

Regression is implemented using bagged decision trees and mean square error; 100 trees are used in each forest. Each training subset is used to predict lung function for a given test instance, resulting in an ensemble of predictions. The final decision is made by clustering the ensemble using k-means (k=2). The centroid of the cluster with the most instances is the final prediction of PEF, FEV_1 , or FVC.

5.4.2.3.3 Spirometry Curve Regression

The shape of the curve is a more difficult and involved regression. Instead of a single measure to regress to for each curve, flow rate and volume for each 15ms frame of the curve needs to be estimated. The ideal regression should use not only the feature value, but also the correlations across time (as flow rates that are close in time should also be close in value). To exploit this, SpiroSmart uses a conditional random field (CRF) [128] and a bagged decision tree regression. In order to reduce the complexity for CRF based regression, normalized flow-volume curves aimed at obtaining the correct shape of the curve is also generated. Once the shape is obtained, the curve is scaled by measures from the lung function regression stage.

This process is then repeated for the volume (*i.e.,* each feature is integrated and the volume curve is regressed to rather than the flow curve). This results in separately calculated flow and volume curves. Much like the regression for lung function measures, the curves from different subsets are clustered using k-means with k=2 (in this case the area under the curve is used to cluster). Before clustering, outliers are removed and curves that are physically impossible are discarded (*i.e.,* the volume is not monotonically increasing). Finally, the average of the curves in the largest cluster is taken as the predicted flow-volume loop.

5.4.3 Results

In this section, I discuss the performance of the initial prototype of SpiroSmart when compared to a clinical spirometer in terms of the accuracy of estimated lung function measures and false positive *vs.* false negative readings. I then discuss the ability of participants to use SpiroSmart without the need for a mouthpiece or sling to control distance. Finally, I discuss the accuracy of the curves SpiroSmart generates and compare different diagnoses from pulmonologists using SpiroSmart and a clinical spirometer. Based on the evaluation I suggest that SpiroSmart has the potential to meet the needs of home lung function monitoring and needs to be tested further.

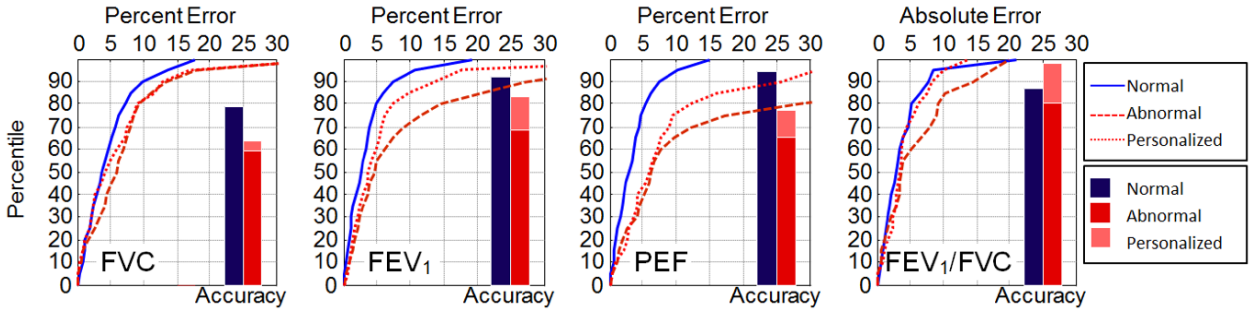

Figure 5.9: Cumulative error plot of the percentage of the results (y-axis) within the percent error of the x-axis, shown for "normal", "abnormal", and "personalized abnormal" groups. Also shown are the accuracies for each measure, defined as the percentage of measures within accepted clinical limits (*i.e.*, the variation one would expect on a traditional spirometer).

5.4.3.1 Estimate of Lung Function Measures

The comparison of measurements from SpiroSmart and a clinical spirometer is broken down by how the percent error is distributed and how well this conforms to accepted clinical variances in each measure.

5.4.3.1.1 Distribution of Percent Error in Lung Function Measures

The curves in Figure 5.9 present the cumulative percentage error plots for FVC, FEV_1 , PEF, and FEV_1/FVC **Description:** The horizontal axis on the top shows the percent error between the actual and predicted value. The vertical axis shows the percentile of the lung function measures that are within that percent error. Hence, curves that approach the top left quickly are more accurate for a greater percentage of subjects. The results are categorized by normal subjects, abnormal subjects, and abnormal subjects whose models have been personalized. **Result:** For all lung function measures the algorithms perform best on normal subjects, and abnormal distributions tend to have longer tails. The mean percent errors are 5.2%, 4.8%, 6.3%, and 4.0% for FVC, FEV_1 , PEF, and FEV_1/FVC, respectively. When personalization is used, the means improve to 5.0%, 3.5%, 4.6% and 3.6%. The personalized models significantly improve FEV_1 , PEF, and FEV_1/FVC for abnormal subjects (based on an F-test of the residual variance, $p < 0.05$), but do not

significantly improve FVC. Personalized models for normal subjects (not shown) are not statistically significant. **Implication:** SpiroSmart produces results that are consistent with other handheld spirometers, even without personalization [186, 226]. The long tails for abnormal subjects require further investigation, but it appears that personalization brings the error distributions much closer to that of normal subjects.

5.4.3.1.2 Accuracy of Lung Function Measures

Bar graphs are also shown in Figure 5.9, displaying the "accuracy" of each measure categorized by normal, abnormal, and personalized, as before. **Description:** For FVC, FEV_1 , and PEF the accuracy is calculated by finding the number of measures that fall within a certain clinically relevant range. A range is used because the "actual value" of the measure is not specifically known. As previously discussed, a subject cannot simultaneously use a spirometer and SpiroSmart, so actual ground truth is unattainable. The range is calculated using ATS criteria for what constitutes a "reproducible" lung function measure [155], and published limits of variability for that measure of lung function [43, 127, 186, 192]. For example, a subject's FVC values can consistently be within 0.05 L or within 7% over short durations [155]. **Result:** From the accuracies, it is apparent that, for normal individuals, FVC is within the range of expected variability almost 80% of the time and FEV_1 and PEF over 90% of the time. However, there is a significant drop in accuracy for patients with abnormal lung function. **Implication:** Most subjects will almost always produce similar FEV1 and PEF whether using SpiroSmart or a spirometer. When abnormality is detected, personalization can be performed at the clinic to increase performance. FVC has the least accuracy and it appears personalization has little effect.

Description: The accuracy of FEV_1/FVC is calculated differently than the other measures. Notice that FEV_1/FVC is already a percentage and the x-axis on top is absolute error (not percent error). FEV_1/FVC is the most common measure of lung function used in diagnosis [213] and there are a number of standards for interpreting the value. For example, if the FEV_1/FVC is less than 80%, the subject is generally considered to have

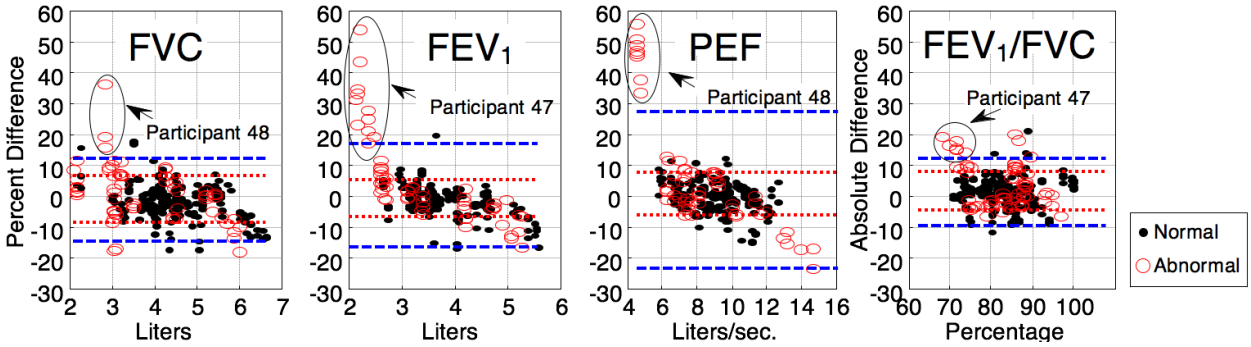

Figure 5.10: Bland Altman plots of percent error between SpiroSmart and a clinical spirometer *vs.* the value obtained from the clinical spirometer. ±2σ (long dashes) and the 10th and 90th percentiles (short dashes) are also shown. Outliers in each plot are the results of two participants (P48 and P49), as shown.

obstructive lung function [213]. If greater than 80%, other measures are used to interpret lung function. The primary motive of this measure is to classify between obstructive, non-obstructive dysfunctional, or normal lungs. Hence I consider estimation accurate (a) if both, spirometer and SpiroSmart estimate FEV_1/FVC values on same side of 80% (either lower or higher); or (b) the two estimates are within ±3 percentage points [155, 213]. **Result:** Both normal and abnormal subjects have similar accuracy (80-90% of all results), and personalization increases this accuracy to near 100%. **Implication:** SpiroSmart can be effectively used to diagnose normal vs. obstructed spirometry using FEV_1/FVC, especially when a personalized model is used.

5.4.3.1.3 False Positives *vs.* False Negatives

In previous analysis, it wasn't investigated whether the lung function measure is under-estimated or over-estimated. **Description:** To elucidate the direction of the bias, Figure 5.10 shows modified Bland-Altman plots of each lung function measure, showing percentage difference between the spirometer and SpiroSmart versus the spirometer measure. Measures taken from normal subjects are shown as black dots and abnormal subjects are circles. Lines indicating the ±2σ (long dashes) and the 10th and 90th percentiles (short dashes) are also shown. **Result:** From these plots it can be seen that SpiroSmart tends to over-estimate actual value for abnormal subjects (a false negative).

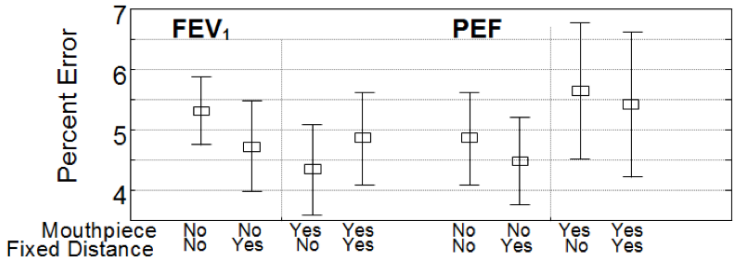

Figure 5.11: Average percent error for each configuration. Errorbars are standard error.

However, the most prominent outliers in each plot are from a single individual, which also accounts for the long tails in Figure 5.9. **Implication:** SpiroSmart generalizes well across normal and abnormal participants. However, the real problem is that for some participants (2 of 52 in our dataset), SpiroSmart gives completely erroneous values. However, for these subjects, SpiroSmart returns values with large dynamic ranges that could be used to detect when a subject is an outlier.

5.4.3.1.4 Confounding Factors and Trends

An 8-way ANOVA was performed to determine if other variables significantly contributed to the magnitude of the residual variance (difference between SpiroSmart and spirometer). I used grouping variables of height, weight, gender, recent illness, if a mouthpiece was used, if distance was controlled using a sling, whether the subject had performed spirometry tests before, and how often the subject used a spirometer. The only significant variable was if the subject had used a spirometer in the past ($p < 0.001$). The level of agreement and reproducibility between spirometry measures is highly correlated with how experienced the subject is at performing the tests. The only other variable suggestive of being associated with bias magnitude (p<0.1) was mouthpiece use.

This is investigated further in Figure 5.11. **Description:** Figure 5.11 shows the average percent error in the lung function measures, categorized by the four different configurations. **Result:** For FVC (not shown), there are no statistically significant differences. For FEV_1 , there is a small but significant decrease in the percent error when both a mouthpiece and sling are used — about 0.5% decrease on average. For PEF, there

is a significant reduction in the percent error when no mouthpiece is used, about 1% on average. This may at first seem surprising, but recall that the measure of flow rate is derived from a model of the lip reverberation — a factor that is diminished when using the mouthpiece. **Implication:** For general spirometry, the mouthpiece and fixed distances are not indicated. However, for clinical studies that report on a specific spirometry measure, a mouthpiece might be indicated for FEV_1 or contraindicated for PEF.

5.4.3.1.5 Participant Feedback

It is interesting that a mouthpiece and fixed distance did not significantly improve measures of lung function. This implies that the subject does not need to carry any additional equipment for performing spirometry outside of the clinic. However, some subjects expressed a preference for using the mouthpiece during the test. In an exit survey, the participants were asked to rate the ease of use and comfort level with each of the configurations. 54% chose the "mouthpiece" only as their strongest preference. Many subjects expressed that it "felt awkward without the mouthpiece" (Subj. 22) or that "I felt like I had to concentrate on the position of my mouth when it was just me and the phone" (Subj. 35). Thus, it may be more comfortable for some individuals to use a mouthpiece during the tests, even though measures of lung function are unaffected. Some participants felt "the phone is so much more mobile without the mouthpiece" (Subj. 43) and "it made the test more simple" (Subj. 20). In these scenarios, carrying a mouthpiece (that also needs to be cleaned) is perceived as an impediment.

5.4.3.1.6 Evaluating Curves Generated by SpiroSmart

To investigate whether the shapes of curves and lung function measures could be used for diagnosis, an online survey was designed for pulmonologists to interpret curves from SpiroSmart. Three subjects with abnormal curves were selected at random and the two abnormal subjects that made up the outliers in Figure 5.10. The outlier subjects were

Table 5.2: Summary of survey responses from 5 pulmonologists. Also shown are the number of times a pulmonologist's dianosis matched (within rater agreement) and were similar to within one degree (within rater one-off).

Curve	Range of Diagnoses		Within Rater	
	Spirometer	SpiroSmart	Agree	One-off
1	Mild	Minimal	3	1
2	Mild	Mild	3	1
3	Nomal	Mild	3	1
4	Normal	Normal	5	0
5	Normal	Normal	5	0
6	Minimal - Moderate	Restrictive	0	0
7	Minimal - Moderate	Mild	0	2
8	Insufficient	Insufficient	3	0
9	Normal	Normal	5	0
10	Normal	Normal	5	0
Total	N/A	N/A	**32/50**	**5/18**

chosen to investigate the worst case performance of SpiroSmart. These and five randomly selected subjects with normal curves (a total of 10 subjects) were used in the survey. SpiroSmart curves from the no mouthpiece, fixed distance configuration were used (again, chosen at random). One representative curve from each system was selected for each subject. Per ATS criteria, curve with the largest FVC value was selected [155].

The curves were compiled into an annotated sheet that pulmonologists regularly see summarizing a spirometry session. Along with the resultant volume *vs.* time and flow *vs.* volume plots, the sheets identified subject's age, gender, height, and weight along with the expected lung function values [119]. The measured FVC, FEV_1 , and FEV_1% were also shown along with the percentage of expected (*i.e.,* measured/expected $\times$ 100). The summary sheet configuration was based on an example from a local clinic. Two digital sheets (from the clinical spirometer and from SpiroSmart) were created for each of the ten subjects.

Five pulmonologists completed the survey over a two-week period. The survey took 20-30 minutes. Four respondents worked primarily in pediatric pulmonology and one in adult pulmonology. Two respondents reported that they use these types of summaries 5-10 times per week, two use them 10-20 times per week, and one used them >20 times per week. The pulmonologists were shown each of the 20 summary sheets in the online survey in random order and were not aware which came from SpiroSmart and which came from the clinical spirometer (they were told that some curves may be from a cell-phone based spirometer).

For the survey, the pulmonologists were asked to view each sheet and select whether the curve suggested normal lung function, obstructed, restrictive, or whether the information presented was insufficient. If obstructed, classification into minimal, mild, moderate, or severe was sought (*i.e.,* a total of 7 possible diagnoses). These categories are the exact categories used on standard spirometry sheets used by pulmonologists. There was also an optional comments block, where the doctors could enter any additional information about the diagnosis.

Description: Table 5.2 summarizes the results of the survey. For each curve, the range of

responses is shown for the spirometer and SpiroSmart. Also shown are the "within rater agreement" (the number of times the ratings from one pulmonologist agreed exactly) and the "within rater, one off" (the number of times ratings from the same pulmonologist are off by one degree). For example, "Mild" and "Moderate" obstructions would be considered one away from each other and so would "Normal" and "Minimal Obstruction". "Restrictive" and "Insufficient" were not considered 'one off' from any other response. Note that curves 6 and 7 are from the subjects whose measures were considered outliers in Figure 5.10. **Result:** Overall, the pulmonologists had strong agreement with one another. For normal curves (3, 4, 5, 9 and 10) there was a general agreement from the pulmonologist regardless of whether they used SpiroSmart or a traditional spirometer. 23 of the 25 responses matched identically and 2 responses were false positives rated as Minimal or Mild obstructions. Curves 1, 2, and 8 are rated with 9 of 15 responses matching exactly and 2 being one degree off. Of the four remaining, three were false positives, indicating worse obstruction or insufficient effort, and one was a false negative (mild obstruction categorized as normal). **Implication:** For these eight subjects the range of diagnoses for each subject is nearly identical. It appears SpiroSmart can be effective for diagnosing not only if the patient is obstructed, but also the degree of the obstruction. More data is still needed from a larger survey and database to establish clinical significance. For the data presented here, the initial results appear very promising.

As expected most disagreement occurred in curves 6 and 7 (the outliers marked in Figure 5.10). **Result:** For curve 6, all respondents thought the curve was restrictive because of a falsely high FEV_1 value. The low FVC indicates the abnormality is not from obstruction, but from restriction. Even so, it is encouraging that the curve is judged as abnormal on both the Spirometer and SpiroSmart. For curve 7, the FEV_1% is falsely high in SpiroSmart, resulting in three of the five responses being false negatives. Implication: In this worst-case scenario, the inflated lung function values caused diagnostic disagreement in pulmonologists, despite the curves having a similar "scooping" shape.

5.4.4 Implications

The evaluation of this initial prototype shows that SpiroSmart has the potential to meet the needs of home lung function measurement. However there are some limitations. First, the setting where the test is performed must remain relatively quiet. This also precludes the coaching that normally occurs during spirometry for children and adolescents. A technician motivates the user with high-energy explanations like ”really push it out” and ”keep going.” With SpiroSmart, these motivations would need to be handled from a user interface or embedded into the playing of a blowing game. I have been working with a user experience researchers and designers and we have been evaluating various interfaces to ease in coaching. These will be discussed in more detail in Section 5.5.2.

Another limitation is that there is a decreased accuracy in FVC compared to other measures (even when personalization is used). This may be a limitation of audible flow rate sensing. Only about the first three seconds of the test is audible. The first two seconds make up 90-100% of the flow-volume loop and includes FEV_1 and PEF (which could explain why these measures are so accurate). However, the remainder of the test is mostly inaudible. Most probably the regression is extrapolating what the FVC should be, rather than regressing from the audible signal. Thus, it is worth exploring if FVC accuracy can be increased by creating an interface that requires users to bring the phone closer to their mouth as the test progresses, and thereby boosting the signal-to-noise ratio at the end of the test (discussed in further detail in Section 5.5.2)

It is unclear from this study how well SpiroSmart will work outside of a lab settings and track trends of a larger population. Subsequent sections will go into more detail of the past and current trials to evaluate SpiroSmart outside of a lab setting.

Given these limitations, it is useful to examine the context and utility of this technology. The current implementation is most useful for replacing home spirometers of individuals with chronic lung ailments. These patients are already familiar with the maneuver, can easily be in a quiet environment somewhere in their home, and likely have access to a

smartphone. Smartphones are, at times, accused of widening the "digital divide" and push people in low socio-economic environments further away from facilities. However, the cost of current home spirometers has already done that. SpiroSmart lowers this cost, helping devices to pervade such environments. As smartphones become cheaper, technologies like SpiroSmart will help in improving access to medical devices and, rather, narrow the digital divide. Agroup that may initially encounter problems adopting SpiroSmart is individuals who have never used a spirometer. Once we better understand ways to train individuals to perform a spirometry maneuver with SpiroSmart, we can lower the adoption- and access-barrier. This can then lead to wider adoption of spirometry in other areas like non-chronic disease management, air quality effects, allergic reactions, *etc.*

5.5 Deployments

The controlled evaluation of the initial SpiroSmart prototype demonstrated that it has the potential to lower the access barrier to spirometry. However to fully evaluate and fulfill that potential it is extremely important that the app is tested with far more (in number as well as diversity.

5.5.1 Sites and Demographics

The initial evaluation had 52 participants and the while there were some mild or moderately obstructed participants (Table 5.1), the system was still not tested for performance and usability on users with more severe ailments. Since the initial prototype in 2012, SpiroSmart has been involved in numerous clinical studies. It is current deployed in multiple locations around the world; including Seattle and Tacoma in USA, Khulna in Bangladesh, and Pune in India. The purpose of these multiple deployments is to get access to diverse user populations. For instance, two major sites that are testing SpiroSmart in the state of Washington in USA are Seattle Children's Hospital and Veterans Affair Hospital in Tacoma. In terms of age, when compared to each other these two sites provide very different user groups (below the age of 15 and above 55,

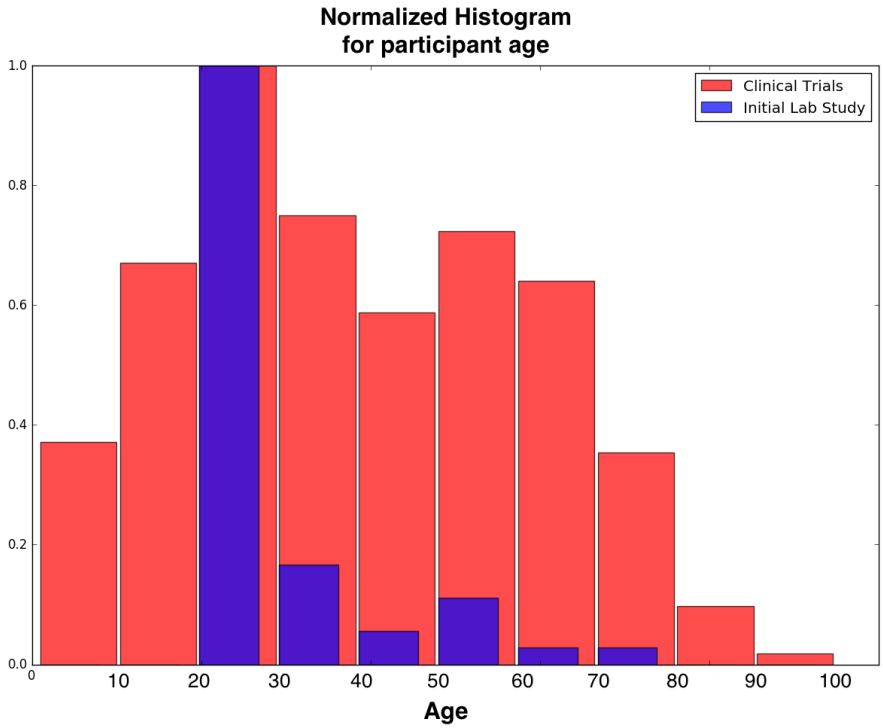

Figure 5.12: Histogram of age distribution of participants in the initial feasibility study and the clinical trials.

respectively). Figure 5.12 provides comparison of histograms of patient age in the initial evaluation and the clinical trials.

Apart from age, degree of lung ailment is another important factor and it is extremely important to get adequate representation from all populations. This is especially true for a system like SpiroSmart, that is built on machine learning models and relies on adequate samples across the spectrum in order to build a model that is not overfitted to a particular population. Figure 5.13 shows comparison of histograms of patient FEV_1% in the initial evaluation and the clinical trials. To date, SpiroSmart has been used by over 5,000 patients. Although at this stage, it is a bit premature to do a deep analysis of the collected data, there are a number of challenges that have come up. These challenges have made sure that the smartphone app keeps evolving and becoming more user-friendly and much more effective clinical data collection tool.

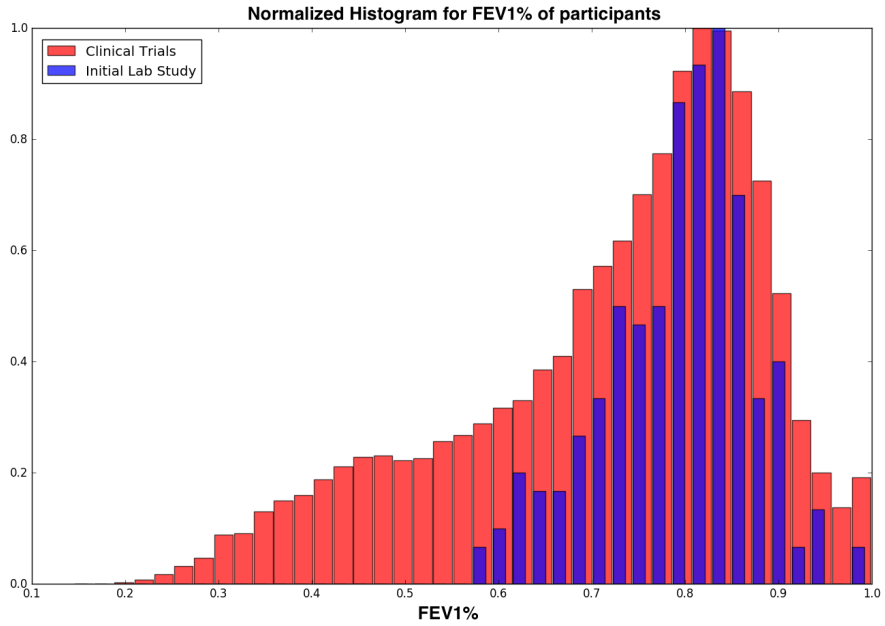

Figure 5.13: Histogram of FEV_1% distribution of participants in the initial feasibility study and the clinical trials.

5.5.2 Challenges Faced and their Mitigation

While solving for the various challenges, SpiroSmart now uses more than just the on-device microphone. It also uses the inertial sensors and the front facing camera. Apart from the use of these other sensors, the user interface has also evolved. I have clustered the challenges raised by the clinical deployments by the nature of their solutions. Some of these solutions are still in very early stage and it is hard to evaluate to what degree will they mitigate the issues. Nevertheless, I believe it is important to understand the kind of challenges a project that uses consumer devices and its sensors can face when deployed in the real world. I provide a glimpse of my experience with such issues from the perspective of SpiroSmart and hope that it will help the reader in developing a better understanding of how to deploy such sensing projects. In the next chapter, I will use these issues and their solutions as the foundation for some of the recommendations on how the sensors and their overall hardware/software architecture needs to evolve to help in effective development and deployment of on-device sensor-based projects.

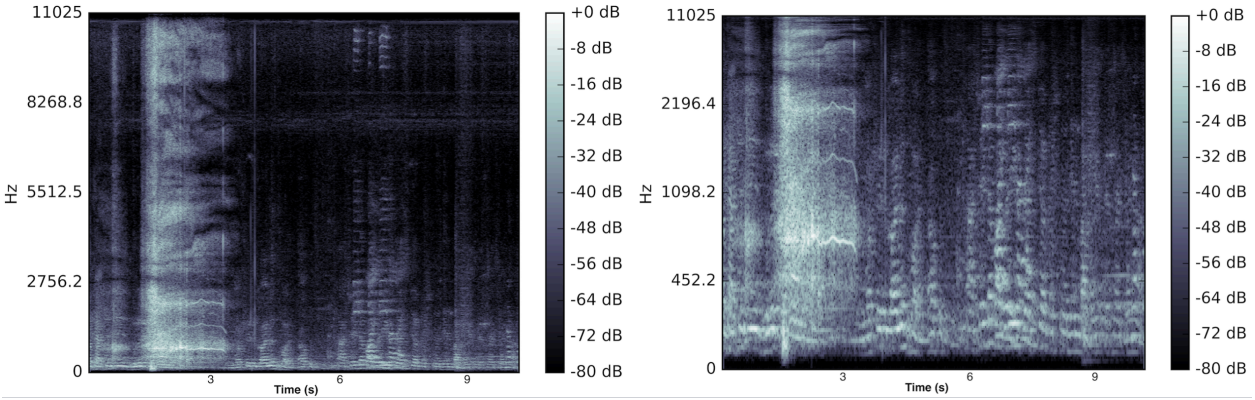

Figure 5.14: SpiroSmart: Spirometry effort with background ambient sound. *(Left)* Time on linear scale. *(Right)* Time on log scale to highlight the speech resonances.

5.5.2.1 Challenges Mitigated through the Microphone

5.5.2.1.1 Ambient Noise

Considering SpiroSmart uses the sound generated by the user's vocal tracts as a proxy of the rate at which they are exhaling, it needs to be used in a somewhat quiet environment. When the app was deployed at different locations, the clinical staff was advised to perform the tests in a "quiet environment". However, it turns out the degree of quietness is not straightforward to establish and enforce on the user. For instance, for some participants the gentle humming sound from an air conditioner or a computer can be too much noise, and for others someone talking gently in the background might be fine as well. Figure 5.14, *Left* shows the spectrogram of a test recorded at one of the test sites and there were people talking in the background during the test. Figure 5.14, *Right* shows the same test on a log time scale to show some of the resonances from the speech and it can be seen that they interfere with the resonances.

Although mel spectrograms and cepstral coefficients (MFCC) can perhaps be used to separate out the speech from the spirometry sounds I haven't tested it sufficiently [240]. Currently, instead of trying to separate the speech out, I simply detect the level of ambient sound and invalidate a test if the ambient sound is above an experimentally determined

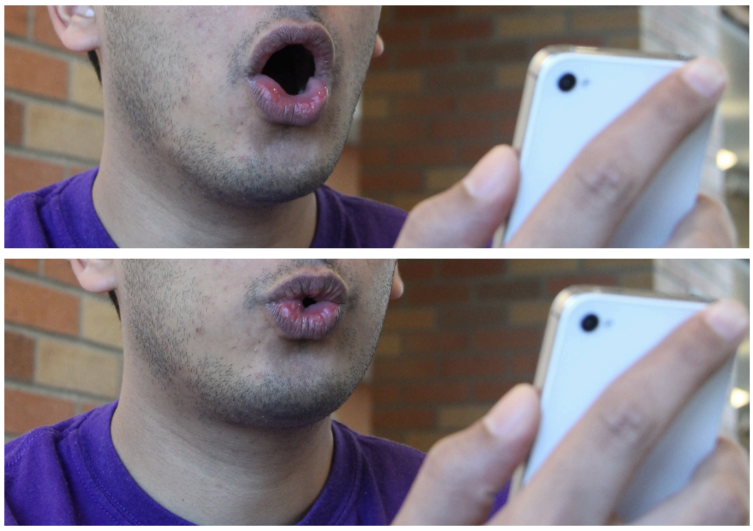

Figure 5.15: ***(Top)*** **Correct lip posture.** ***(Right) Pursed Lips.***

threshold. When the patient performs the spirometry effort, they first inhale as much air as they possibly can, this part of the test does not generate any sound. So SpiroSmart uses the inhalation phase to monitor the ambient sound level and it is too loud, it prompts the user to find a quieter setting.

Overall, the issue of ambient noise raises a very important issue with generic sensors, such as microphones and cameras. These sensors are very sensitive to things around us. They listen/see everything and that makes them very powerful and vulnerable. It will always remain very important for systems built on top of these sensors to remain resilient ambient noise. Although the solution suggested in this section works for SpiroSmart, it will not necessarily work for all applications. Therefore the main takeaway here is to keep this challenge in mind and make sure to overcome it when deploying the applications in a real-world setting.

5.5.2.1.2 Lip Posture

Traditional spirometers require users to exhale through a mouthpiece. It serves two main purposes: (1) It keeps the device clean to be used by other users, and more importantly

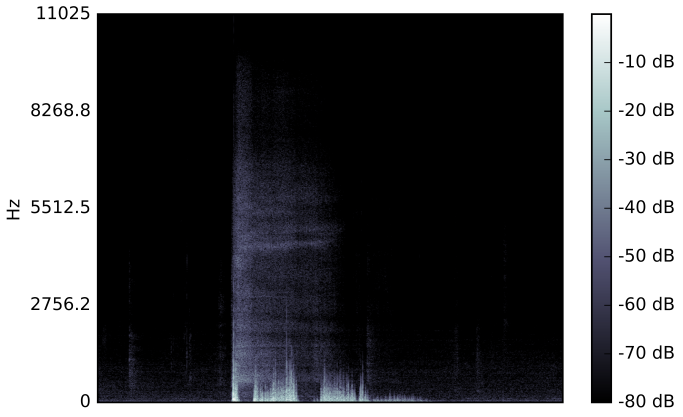

Figure 5.16: Spectrogram of a spirometry effort with pursed lips.

(2) it channels the air through a tube of known dimensions. SpiroSmart does not necessarily require users to exhale through a mouthpiece; they can use it if they desire. Not using the mouthpiece means that the participants need to main their lip posture and make sure that their mouth is open enough and no impedance is created at their lips and air flows unobstructed (Figure 5.15). Most users need to be reminded about this requirement initially as there is a general tendency for them to purse the lips while exhaling. While this has not been a major issue in the clinical trials as the technician can always explain the user that they need to keep their mouth open. However, I envision it being a major issue when the users use SpiroSmart at home, away from the clinical supervision. Currently I use the audio data itself to detect if the participant pursed their lips. Figures 5.16 and Figure 5.17 show the audio spectrogram of spirometry effort by the same user with their lips pursed and then with their mouths open. It is clear from the spectrogram that one can look for the presence of resonances from the vocal tract to detect if the patient kept their mouths open. This is fairly robust because if the lips are pursed, the flow rate is low enough that the vocal tracts won't resonate. However, this capability still needs to be tested more rigorously in the clinical trials.

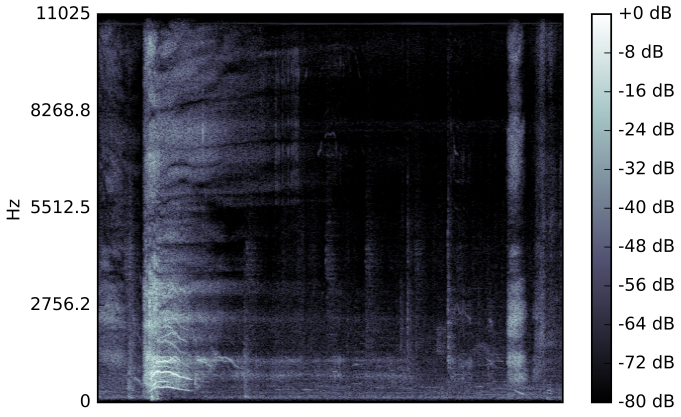

Figure 5.17: Spectrogram of a spirometry effort by the same patient as in Figure 5.16 with mouth open

5.5.2.1.3 Occluded Microphone

Another issue that arises quite frequently is when the user accidentally put their finger in front of the microphone and it gets occluded. This results in an attenuated recorded signal that can be erroneously inferred as low lung function by the SpiroSmart algorithm. These kind of issues are more common with applications like SpiroSmart because they there is a lot of flexibility in how a sensor can be used. For example, when a user uses a clinical spirometer, they need to make sure that they use a mouthpiece and then they exhale into the device and nothing leaks from the sides. However, the level of unpredictability in much higher with SpiroSmart and the occluded microphone is just one of the many issues.

In order to detect if the microphone is occluded either by the user's fingers or may be some lint/dust deposition has happened, the app frequently runs diagnostic checks. The app would randomly emit a 18 kHz tone during the test and then it keeps track of the magnitude of the pitch at 18 kHz. This won't stay consistent across days because the reflections in the environment can be a big confounding factor and it is definitely not foolproof. However, it still helps in removing some obvious erroneous cases. Assuming that the reflection profile of the room won't change much within these consecutive tests, the app can notify the user to make sure that the microphone is not occluded.

5.5.2.1.4 Variation in Sound Levels

The raw magnitude of the sound generated by a user will depend heavily on their height. The reader might recall from Section 5.4.2.2 that the peak magnitude of sound is a strong estimator of the maximum flowrate achieved by the subject during a forced expiratory maneuver (PEF). Moreover, normal predicted PEF of a patient is given by [83].

PEF depends heavily on the participant height and if for a moment we assume that up to a certain point, the user's height will be a function of their age (and many other factors), then we can see from Figure 5.12 that the diversity in age is much higher in the clinical trials, as compared to the initial controlled study. This means that the users of the app will generate very different levels of sounds while performing spirometry. The initial prototype of the app used some experimentally determined thresholds to detect the start of the tests. However, with an expanded diversity in sound levels, those thresholds were made dynamic. The patient's demographic information was used to calculate their predicted PEF and then that was regressed back to an estimated sound level (using the already collected data) and then a 30% of that sound level was used as a threshold to detect the onset of a spirometry test. While this improved the performance for a significant number of users, there were still many that were unable to use the app because the amplitude of sound for them was too low. This required us to make changes to the usage protocol of the application and I will discuss those in more detail in the next section (Section 5.5.2.2.1).

5.5.2.2 Challenges Mitigated through Other Sensors

5.5.2.2.1 Variation in Sound Levels

I wanted to come back this issue of variability of sound levels and discuss it from the perspective of the cameras as well. Considering that different users will produce different magnitude of sound, as the sound level goes down, so does the signal to noise ratio. This problem is compounded further by the fact that for a significant portion of the spirometry test, the generated sound is very low in magnitude. Therefore, after initial

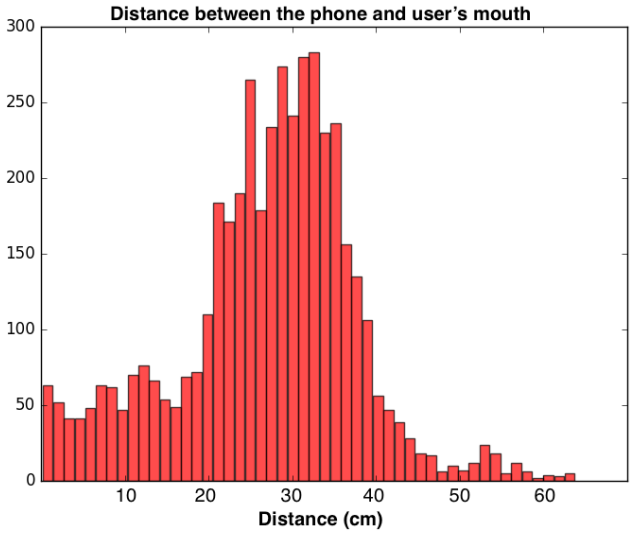

Figure 5.18: Distribution of distances at which participants put the phones while using SpiroSmart in clinics.

stages of the clinical trials with 200 subjects, I added the option to do the test at varying distances. This was in contrast with the initial prototype and its evaluation discussed earlier (Figure 5.6), where we went as far as fixing the distance between the phone and the camera with a sling. As the variability in the sound level increased as the patients started using phone at different distances (Figure 5.18) shows the distribution of distances across participants, it became important to calculate the distance automatically.

I use the phone's front facing camera to calculate the distance to the user's mouth. The initial approach was to detect the user's eyes and then calculate the distance between the two eyes as an inverse proxy of distance between the phone and the user's mouth. However, this needed to be tested and tuned rigorously before deployed in the field. As an intermediate step, the app currently ask the participants to put a business card (usually of standard size) in front of their face, take a photo with SpiroSmart app, and then draw a box around the card manually. This manual steps makes the process less seamless but it also is less noisy. The plan is to use this intermediate step as a data collection opportunity and think of the distance between the eyes as a linear function of the size of the card. One can easily imagine having a regression function that converts between the two values and the manually recorded data that has been collected on more than 3000 patients can prove

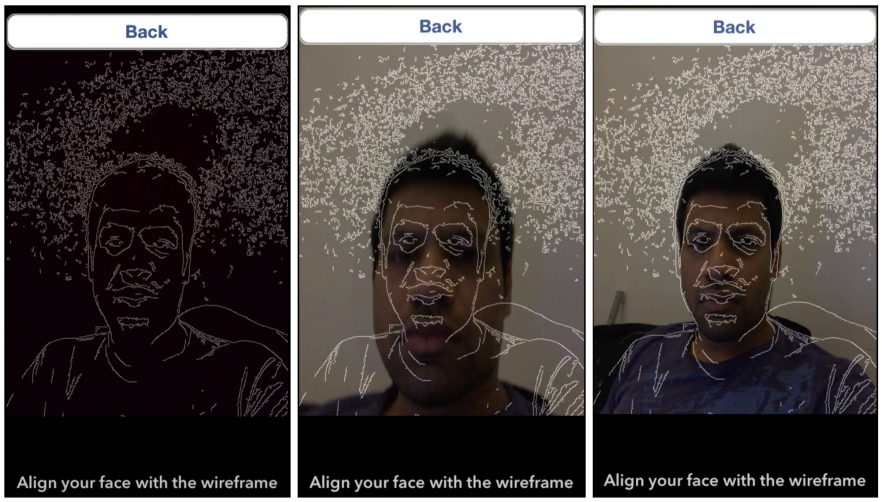

Figure 5.19: SpiroSmart uses the front-facing camera to make sure that the participants perform tests at a consistent distance. Once the participant has set a distance that works for them, for each subsequent effort the app shows them the silhouette of these face so that they can align themselves correctly.

Figure 5.20: Participant using the application at a wrong angle. The app uses the gyroscope to check for such mistakes and notifies the user.

extremely valuable. Also, when the same patient comes back for a trial, the app shows them a silhouette of their face from the last study to remind them of the optimum distance for them to avoid too much distance variability for the same patient (Figure 5.22.

5.5.2.2.2 Phone not Held Properly

This is again one of those issues that are not too troubling in the clinical trials as there is a clinician to train the users. However I envision this being a big issue when the users use the

Table 5.3: Reasons for rejection of spirometry efforts (as selected by the technician supervising the session)

Reason for effort rejection	Number of tests
Reduced Effort (n, %)	3914, 8.7%
Short Effort (n, %)	3234, 7.1%
Cough (n, %)	11108, 24.7%
Noise (n, %)	8047, 17.9%
Phone too close (n, %)	91, 0.2%
Phone too far (n, %)	34, 0.001%

app at home. Sometimes participants hold the phone at a very different angle from desired (Figure 5.20). Currently the app uses the on-device gyroscope to calculate the angle of the phone and it notifies the user whenever the angle is too off. I am also working with designers and we are involved in a study to evaluate what are the best ways to introduce and train the users on the user-facing version of SpiroSmart. I believe a combination of descriptive user interface and intelligent use of the on-device sensors should be enough to train the users on how to use SpiroSmart. However, again this needs to be tested rigorously in the field before we can be absolutely certain.

5.5.2.3 Challenges Mitigated through UI and Data Collection

A number of issues that have been discussed in this section can benefit immensely from a large scale data collection of the valid as well as erroneous uses of SpiroSmart. Sensing that there might be an opportunity to make a machine learning model to automatically rate efforts for validity and quality, I added a quality and validity annotation capability to the app. Considering the current scale of SpiroSmart's deployment, it can be invaluable to have such an annotated dataset.

Machine learning models are generally garbage in – garbage out, *i.e.*, the quality of the

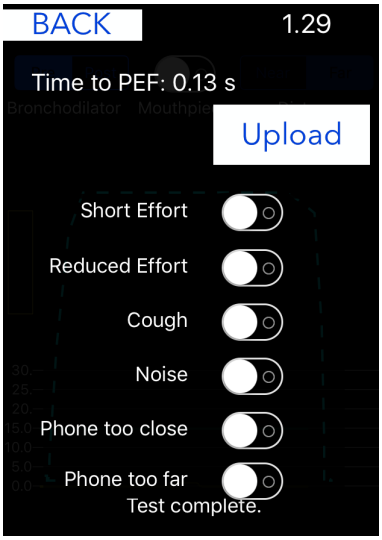

Figure 5.21: The user can annotate why they are rejecting a particular spirometry effort.

output is heavily dependent on the quality of the input. Moreover, considering spirometry is an effort dependent test and the patient needs to be trained to perform a valid test, there are many failed/invalid tests. The clinician walks the patient through the procedure, demonstrates how the effort is done. On average, patients take 2-3 trials before they completely understand how to perform the spirometry effort. Instead of throwing away the invalid trials, the app asks the user to save those efforts as well and it asks them to annotate them with the reason for failure (Figure 5.21 shows the UI). I believe that with enough data, one can build a machine learning model that is able to automatically classify efforts into the classes shown in Figure 5.21. Such a capability can prove invaluable when the app is used by the patients at their homes, away from the clinical supervision. Table 5.3 shows the number of such annotated samples currently collected in the SpiroSmart database.

Moreover, validity is not the only concern. The app also asks the clinician to rate every valid effort on a scale of 1-3 as well. Figure 5.2 shows the corresponding UI. The idea is that the boundary of a valid spirometry effort with SpiroSmart is a bit fuzzy right now. As the algorithm develops a better understanding of the user's physiology and aims to maximize of the accuracy of lung function measures, it is important to use the best

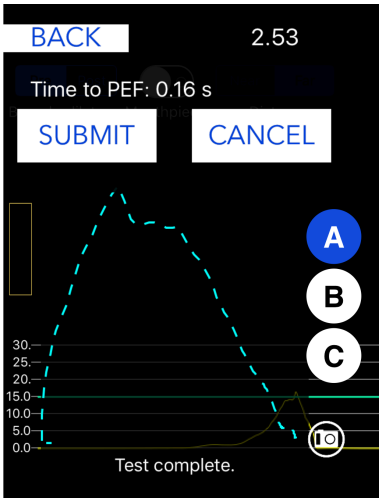

Figure 5.22: The clinician can rate the efforts by the patient on their discretion. This can be used later for building machine learning models to predict test quality.

dataset for training. Currently the best judge of the quality of the effort is the clinician but we won't always have the clinician with every device that is running SpiroSmart. Therefore it will become important for SpiroSmart to self-rate the efforts and all of the collected data and their quality annotation by the clinician might prove extremely useful.

5.6 SpiroCall

In this section I discuss *SpiroCall*, a call-in service that measures lung function on *any* mobile phone, and not just a smartphone. It removes the need for a locally running application. Unlike SpiroSmart, it transmits the collected audio using the standard voice telephony channel. The server receives the data of the degraded audio quality and calculates relevant lung function measures and report them back to the patients using audio or text message. The ability of use a server to analyze the audio data transmitted from any mobile phone, be it a feature phone or smartphone, eliminates the need to develop a specialized application for every phone platform. SpiroCall combines multiple regression algorithms to provide reliable lung function estimates despite the degraded audio quality over a voice communication channel.

SpiroCall was evaluated in a controlled study with 50 patients. I compared SpiroCall to two FDA-cleared spirometers and evaluated the effect of using the voice communication channel on the performance of SpiroCall. Each patient performed spirometry efforts on two different phones recording the audio through the cell phone network and two smartphones recording the audio locally through an app. The results show that SpiroCall has a mean error of 7.2% for the four major lung function measures. For FEV_1% , the mean error is 6.2%.

5.6.1 Motivation

One of the lessons we learned from the SpiroSmart deployments in USA, India, and Bangladesh was that a significant user population does not have a smartphone and those who have one, do not necessarily have the phones of the same kind. This means that even if we focus just on smartphones, we need to develop applications for all kinds of phones and also those that generalize across software architectures and hardware configurations.

Moreover, while smartphone are becoming prevalent at a breathtaking rate, more than half of the mobile phone users in sub-Saharan Africa and South Asia will still be using a non-smartphone (or feature phone) in 2020 [62]. A major portion of the population suffering from lung impairments lives in these low resource environments. In fact, according to a recent WHO report, more than 90% COPD deaths occur in low and middle-income countries [238]. Thus, I believe that phone-based spirometers need to work on all mobile phones, and not just programmable smartphones. Even within smartphones, the diversity of phone manufacturers and models makes it challenging to manage custom applications for every type of mobile phone.

5.6.2 Design and Algorithm

SpiroSmart offloads a significant chunk of computation to a server, with the audio transferred via an Internet connection. Thus, the received audio is lossless and free of artifacts. In SpiroCall, I leverage the voice communication channel to transmit the audio

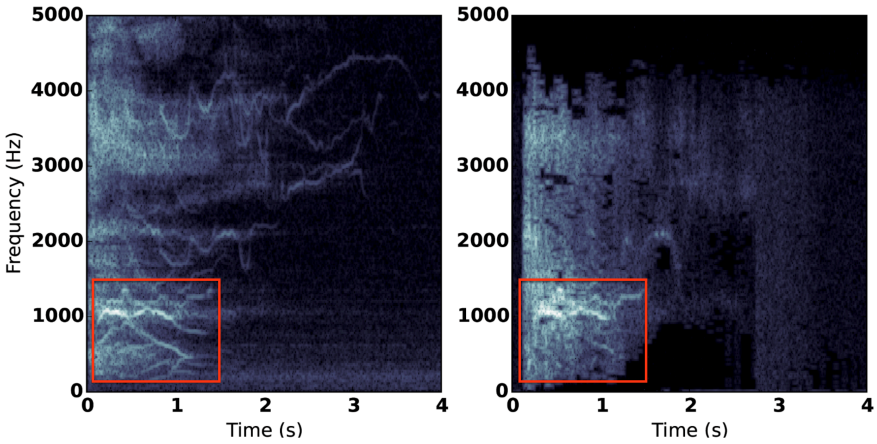

Figure 5.23: Comparison of spectrograms recorded for SpiroSmart and SpiroCall. *(Left)* Spectrogram of a spirometry effort recorded locally, and *(Right)* recorded through the voice channel. The GSM network downsamples the audio, and the data over 4 kHz is lost. However, the data in our main region of interest (red square) is largely preserved and reconstructable.

data to a server. The server then uses machine learning to compute lung function measures. The features used by our machine learning model fall into same three categories as SpiroSmart: temporal envelope detection, spectrogram processing, and linear predictive coding (LPC). The cellphone channel (GSM) uses LPC to encode voice. This means that even though the GSM channel compresses the audio signal, the values of LPC coefficients remain largely preserved. Landlines, or POTS (Plain Old Telephone Service) is also an attractive option as a communication channel. However, I haven't evaluated SpiroCall's performance of POTS yet, because GSM networks are far more prevalent than landlines in the developing world.

Figure 5.23 shows spectrograms of a spirometry effort recorded locally on a smartphone *(Left)* and the same effort after the data is sent through a GSM cell phone network *(Right)*. The device type is an Apple iPhone 4S in both cases. There are significant differences between the two spectrograms as the voice undergoes many changes as it goes through a communication channel. Although different communication networks use different speech coding techniques, all GSM/UMTS speech-coding algorithms share similarities in their treatment of speech and are based upon the same underlying linear prediction approach.

First, all GSM voice coding technologies use a source-filter model for speech. That is, the "source" estimates the lung or glottis excitation, and the "filter" estimates how the vocal tract blurs this excitation into continuous sound. Parameters of the source and filter are then transmitted through the channel, instead of the raw audio. The most common method for separating out the source excitation from the vocal tract filter is to use LPC. An artifact of the LPC calculation is that the strong frequency resonances are preserved (and are calculable directly from the LPC coefficients). These resonances are also the primary features in SpiroCall's algorithms. As such, the LPC encoding is expected to preserve much of the important information in the signal. An example of this is shown in Figure 5.23 – the fundamental resonance is easily seen in both recordings (inside the red box), despite many smaller details, such as higher harmonics of the fundamental resonance and all spectral energy above 4 kHz, being lost.

Additionally, the transmission process suppresses low-energy components in the signal, as can be seen in Figure 5.23 *(Right)* where the energy of the signal abruptly cuts off in patches. In contrast, the signal maintains high fidelity and stays above the noise floor for the initial (and relatively louder) segment of the effort (inside the red box in Figure 5.23, *Right*).

5.6.2.1 Algorithm

In order to deal with the drastic variation in sound quality as the data goes through a GSM channel, I sought to evaluate what modifications are necessary to the algorithm proposed in the original SpiroSmart algorithm discussed in Section 5.4.2.

SpiroSmart and SpiroCall use the microphone as an uncalibrated pressure sensor and the received pressure values are transformed using three approaches (Figure 5.24): (1) envelope detection, (2) resonance tracking in the frequency domain, and (3) linear predictive coding (LPC). The envelope of the signal can be assumed to be a reasonable approximation of the flow rate because it is a measure of the overall signal power (or amplitude) at low frequencies. In the frequency domain, resonances can be assumed to be amplitudes excited by reflections in the vocal tract and mouth opening—and therefore

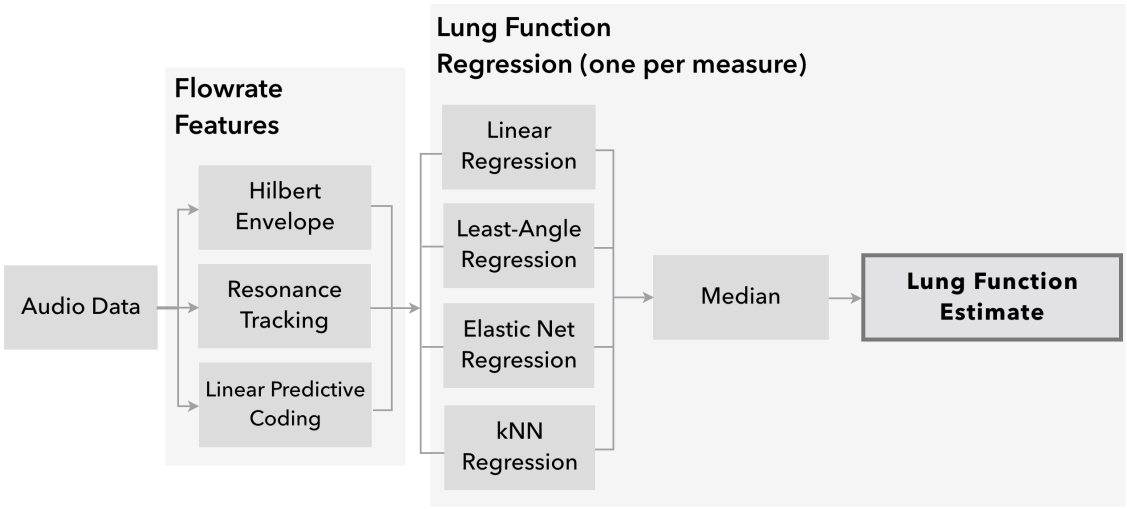

Figure 5.24: SpiroCall lung function estimate regression flowchart.

should be proportional to the flow rate that causes them. Finally, SpiroCall uses linear prediction as a flow approximation. Linear prediction assumes that a signal can be divided into a source and a shaping filter and it estimates the source power and shaping filter coefficients. The "filter" approximates the vocal tract. The "source variance" is an estimate of the white noise process exciting the vocal tract filter - this approximates the power of the flow rate from the lungs. Each approach generates multiple time-domain flow-rate estimations.

I extract separate feature sets for FEV_1 , FVC, and PEF from these time-domain flow-rate estimations. For example, PEF is defined as the maximum flow reached in an effort. Thus, for a given flow-rate estimation, I take the max value and use it as a feature for PEF regression. In contrast, FVC is defined as the total volume of air exhaled. Thus, integrating the flow-rate estimation with respect to time gives us a feature for FVC regression. Using this approach, SpiroCall generates 3 sets of of 38 features for FEV_1 , FVC, and PEF, each. It does not use any regression algorithm to estimate $FEV_1\%$. This value is simply a ratio of the estimated FEV_1 and FVC.

Considering that the GSM channel uses LPC to encode sound, the LPC-based features used in the algorithms here remain largely preserved. The envelope detection-based

features are based on the coarse amplitude of sound with respect to time; in most cases these features remain preserved as well. The spectral features are most affected by the GSM channel because the high frequency details are completely lost. However, upon analysis it was found that the resonances within the first harmonics were strong enough that most spectral features contained some relevant information.

In the original SpiroSmart algorithm, the calculated features were sent to a random forest regression that, because it has no underlying linear model, had trouble exploiting some of the linearity in the feature data. The algorithm performed poorly on the data collected through the GSM channel and it over-estimated the lung function for participants with obstructed lungs ($FEV_1\% < 0.8$). Therefore, the algorithm was updated to employ an ensemble of four different regression algorithms (Figure 5.24), with the aim that each regression would provide a different perspective. The regressions were implemented using the *scikit-learn*[4] toolkit in Python and used leave-one-patient-out cross-validation to avoid overfitting. Furthermore, all the parameters for all the algorithms were kept at their default values and do not tune them for the collected data. (To avoid unintentional manual overfitting on validation data)

The first regression is a linear regression that tries to find a linear relationship between the features and the ground truth lung function value. The second regression uses least angle regression (LARS) [59]. LARS selects the most useful features using a variant of forward feature selection, but the underlying model is assumed to be linear. The third regression uses the elastic net algorithm [250], which eliminates features in a slightly different way than LARS. This regression uses a combination of LASSO regression and ridge regression for regularization that is often more stable. Finally I use enclosing k-Nearest Neighbor regression ($k = 2$) [80], which finds the convex hull of the data in the feature space and fits a locally linear regression. Though the underlying model is assumed linear, the local fitting often can fit many different types of nonlinearity. The final regression estimate is calculated by taking the median of these four regressions. Same process is repeated for FEV_1 , FVC,

[4]http://scikit-learn.org/

Table 5.4: Demographics information for the SpiroCall evaluation participants

Participant Demographics (N = 50)	
Males (n, %)	30 (60%)
Age (yrs) (mean, range)	30 (21 - 67)
Height (cm) (mean, range)	172 (155 - 188)
Reported Lung Ailments: **Asthma: 10 (20%), Bronchitis: 2 (4%), COPD: 2 (4%)** **Cystic Fibrosis: 1 (2%), Sarcoidosis: 1 (2%)**	
Low Lung Function (n, %)	16 (32%)
Never Performed Spirometry (n, %)	30 (60%)

and PEF measures. As mentioned, FEV_1% is calculated as a ratio of the estimated FEV_1 and FVC values.

Additionally, there are situations when the test is performed in a noisy environment or the channel itself might be noisy. To deal with such situations, the system automatically detects the level of background noise by looking at the mean absolute amplitude of the recorded sound for a 250 ms window immediately before the user exhales. This is the period when the user is most silent and the algorithm uses it as an opportunity to measure the ambient noise level. If the amplitude of sound within this window is estimated to be above an empirically determined threshold, the environment is considered unsuitable for data collection. The threshold used here is same as the one used in the SpiroSmart clinical trials.

5.6.3 Evaluation

To evaluate SpiroCall, another extensive dataset of audio samples and ground truth spirometry data was created. I recruited 50 participants (30 males, 20 females), ranging in age from 21 to 67 years (M = 30) through flyers and email messages in the university. The study sessions were conducted in a non-clinical lab setting and lasted for approximately

30 minutes. 20% of participants had mild to moderate lung obstruction, *i.e.,* $FEV_1\% < 0.80$ (Table 5.4). The SpiroCall study used a within-subjects 2×2×3 factorial design. The factors and levels were:

- ***Phone Type:*** *iPhone* and *non-iPhone*.I used two non-iPhone devices: Samsung Note 3 and Sony Ericsson W580i. I used the W580i (feature phone) to evaluate the performance of SpiroCall on an approximately 10-year-old device.
- ***Channel Type:*** *Local recording* and *voice channel recording*. The iPhone was kept consistent in both channels to analyze the performance of SpiroCall if only the channel is changed.
- ***Whistle:*** *No whistle*, *small whistle*, and *big whistle*. This factor will be discussed later when I introduce the vortex whistle in Section 5.7.

All the conditions were counterbalanced and we randomized the order of the whistles.

5.6.3.1 Experimental Setup

I collected the audio data on four phones, two Apple iPhone 4S smartphones, a Samsung Galaxy Note 3, and a Sony Ericsson W580i feature phone. All four phones were in front of the user at roughly an arm's length away (Figure 5.25). The distance was not formally controlled or varied. One of the iPhones and the Samsung Note recorded the audio data locally at 32 kHz and 44.1 kHz, respectively. The other two devices sent the data over the GSM voice channel. These phones placed phone calls to different Google Voice accounts that recorded the data in the form of voicemail messages. Google Voice saved the audio data as 44.1 kHz MP3 files, but the GSM channel band-limited the data to less than 8 kHz. The difference between the local recordings and those done over the GSM channel is shown in Figure 5.23.

The data was transferred from the local phones (Apple iPhone 4S and Samsung Note 3) to the computer over a USB connection at the end of the study. I downloaded the data from the Google Voice accounts as MP3 files to a computer.

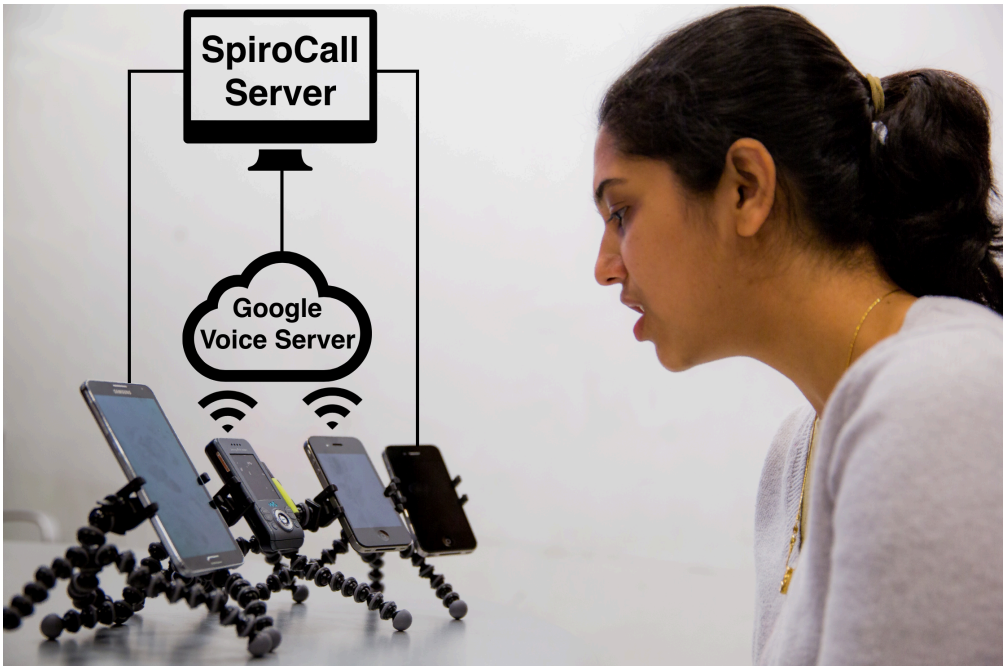

Figure 5.25: SpiroCall experimental setup. The data was recorded on four phones at the same time. Two phones recorded the audio locally. The other two phones called Google Voice numbers and sent the audio data over the GSM channel to the Google Voice server.

5.6.3.2 Procedure

The groundtruth data for the participants was collected on two FDA-approved clinical spirometers: the nSpire Koko Legend and the NDD EasyWare spirometer. I used the two spirometers to answer two questions: (1) whether the participants got fatigued as the session progressed, and (2) how much variability exists between the outputs of the two devices. I recorded the variability between the clinical devices to use it as a benchmark for SpiroCall's performance. The participants performed at least 15 spirometry efforts (three each for: two clinical spirometers, two whistles, and one without whistle). Spirometry measurements are completely effort-dependent and some fatigue can build up when performing this many efforts. Therefore, I recorded efforts on one clinical spirometer at the beginning of the session and on another spirometer at the end of the session. I randomized the order for each participant.

At the start of each session, the participants were explained the forced expiratory maneuver

and were asked to practice using the spirometer. Once the participants were able to perform an acceptable maneuver according to the ATS criteria for reproducibility [155], three efforts were recorded using the spirometer. Next, the participants were introduced to SpiroCall.

The four phones (Phone Type × Channel Type) recorded the audio simultaneously, thus saving the participants from performing tests with each device type separately. I gave feedback to the participants regarding the acceptability and quality of the efforts. In the future, I envision to have a system that automatically determines if an effort was too low in volume and has intelligent user interface that guides the user through the maneuver.

5.6.4 Results

In this section, I discuss the performance of SpiroCall when compared to the two clinical spirometers in terms of accuracy of estimated lung function measures and false positives vs. false negatives. We consider an estimate to be a false negative if the groundtruth FEV_1% is below 0.8 and SpiroCall predicts the value to be above 0.8 [47]. I break down these results by Phone Type and Channel Type. Finally, I discuss the accuracy and usefulness of the flow-volume curves generated by SpiroCall. Based on this evaluation I conclude that SpiroCall can help in screening and monitoring patients with lung impairments in low resource regions.

5.6.4.1 Two Ground Truth Devices

As mentioned, I used two clinical spirometers to collect groundtruth. I compared their respective lung function measure and found that PEF had the maximum difference of 9.2% between the two devices, and FEV_1 , FVC, and FEV_1% had a difference of 5.1%, 5.2%, and 3.2%, respectively. However, none of these differences are statistically significant (based on an F-test, p>0.05). I also studied the effect of order to understand if fatigue played any role in exaggerating the difference between the two devices. In a 2-way ANOVA test with *presentation order* of the two spirometers as a between-subjects factor, it was found that the difference in estimates of PEF and FEV_1 were statistically significant

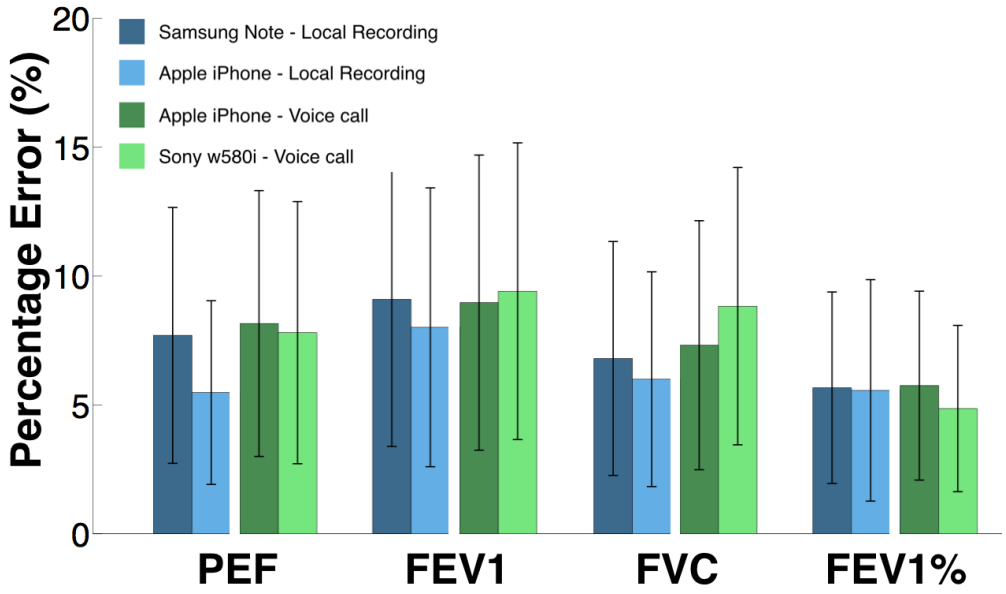

Figure 5.26: Percent error for different lung function measures on different devices. The first two devices recorded the data locally in an app; the next two devices recorded the data over a phone call. The error bars show standard deviation.

($p < 0.05$). This finding suggests that the participants got fatigued by the time the session ended. Therefore, I use the results from the *first* spirometer that the participants used as their groundtruth or reference device. While this means that the reference device was not consistent across participants, the difference in device performance was not found to be significant and should not strongly affect the final analysis. In addition, I corrected for fatigue by counter-balancing between all Phone, Channel, and Whistle Type conditions for all participants.

5.6.4.2 Accuracy of Lung Function Measures

The bar graphs in Figure 5.26 present the percentage error of each measure without a whistle. For all lung function measures, the algorithm returns an average error of less than 10%. There is no significant difference between the performance of smartphones recording the data locally in an app (Samsung Note and Apple iPhone) and phones running over the voice communication channel (Sony W580i and Apple iPhone 4S). The performance of phones collecting audio samples locally can be thought of as running

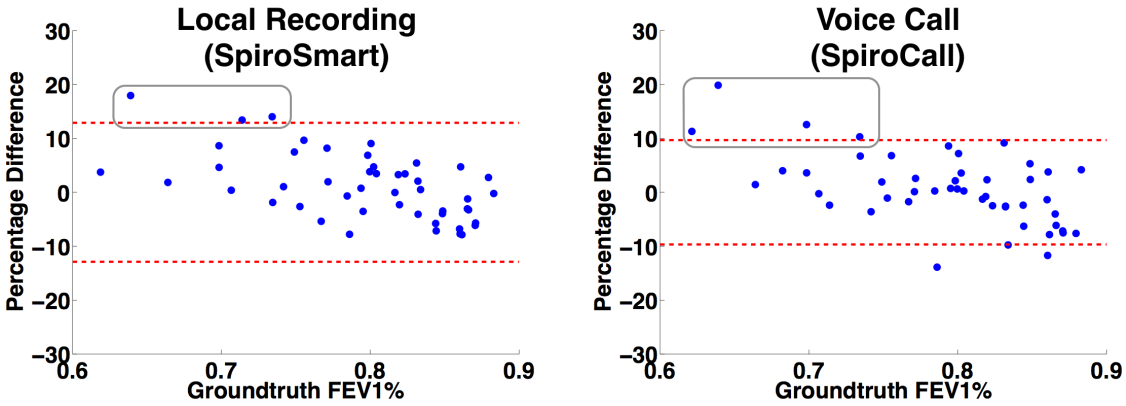

Figure 5.27: Bland-Altman plots of percent error of FEV_1% for local and voice call recordings (SpiroSmart and SpiroCall, respectively) *vs.* the values obtained from the clinical spirometer. The false negatives are highlighted inside the gray boxes. ±2σ (red dashes) are also shown.

SpiroSmart and the ones recording data over the voice channel as running SpiroCall. The performance is best for FEV_1% , which is the most common measure of lung function used in diagnosis because it is typically most consistent [155, 192]. The mean error rate for FEV_1% is below 6% for all the four conditions. The ATS acceptability criteria require lung function measures to be within 7% to 10% of one another [192]. For most patients, SpiroCall performs well within the expected level of variation, even if the patient did not have a smartphone and performed the test on a phone call. However, it is important to evaluate the outliers (with error higher than twice the standard deviation) and see whether the lung function measures are under-estimated or over-estimated. I use twice the standard deviation because the first standard deviation is within the ATS guidelines and the result cannot be considered an outlier.

5.6.4.3 Outliers and Patients with Low Lung Function

In order to understand the direction of the bias, I use the modified Bland-Altman plots [1] in Figure 5.27. The figure shows the percentage difference between SpiroCall and the output of a spirometer versus the spirometer measurement of FEV_1% . Lines indicating ±2σ (red dashes). The focus is solely on FEV_1% because it is the most common lung function measure for diagnosis. If the percentage difference is positive, then the lung function was

over-estimated (false negative). It can be seen in Figure 5.27 that SpiroCall tends to over-estimate the actual value for some patients with low lung function ($FEV_1\% < 0.8$)*i.e.,* a false negative. We highlight the false negatives inside the gray boxes. In any medical device, it is more acceptable to have a false positive than a false negative. The main reason the system currently has more false negatives for low lung function is because the algorithm is data driven and the population with higher lung function is better represented. Therefore, the model tends to bias towards the median value. Considering the signal-to-noise ratio is lower for the devices connected over the GSM channel, the false negatives are slightly more pronounced in case of voice call (Figure 5.27, *Right*).

One way to quantify the model's bias towards higher lung function is to calculate the statistical effect of lung function measure ($FEV_1\%$ in this case) on the error of the model. I tested for effects of groundtruth $FEV_1\%$ on the percent error through a chi-square test. I found that there was a significant effect of the groundtruth $FEV_1\%$ on the accuracy of SpiroCall (p<0.05). As such, the performance of SpiroCall might degrade further if tested on more highly obstructed patients. Although the bias is only slight and there are relatively few false negatives, from a diagnostic perspective, it could mean that patients are screened improperly. This needs to be tested further and evaluated in clinical trials and it needs to be made sure that patients with obstruction are not missed.

5.6.4.4 Curves Generated by SpiroCall

Let us now shift the discussion from lung function measures to the shape of the flow-volume curves. The spirometry curves serve two purposes: (1) to evaluate if the patients performed the effort sufficiently, and (2) to help in diagnosis by showing the descending limb of the FV curve. A technician looks at the slope of the FV curve from the start of the test to PEF. This slope should be as steep as possible, indicating that the initial blast of air was truly maximal. The investigator also looks to see if the user coughs during the spirometry maneuver. Coughing makes the descending edge of the FV curve non-monotonic as the user ends up inhaling during a cough. Therefore, it is important to evaluate how SpiroCall performs in generating these curves.

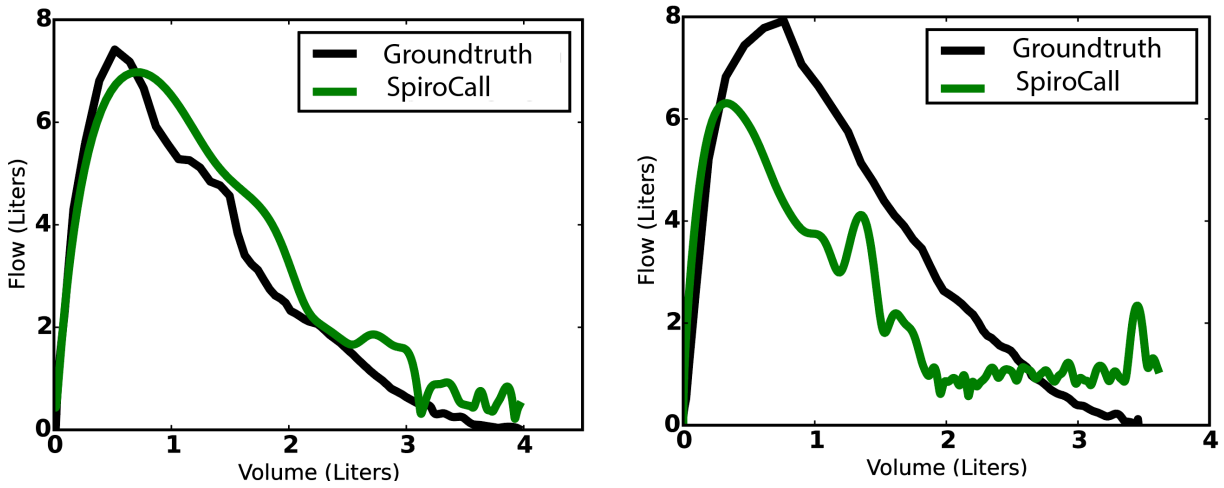

Figure 5.28: Two Flow *vs.* Volume curves generated by SpiroCall

Figure 5.28 shows example flow-volume curves generated by SpiroCall; and these curves can be unreliable. The SpiroCall curve in the Figure 5.28 *(Right)* has an inaccurate shape because the latter half of the effort by the patient was very quiet. When the GSM channel compressed the audio, this segment was heavily compressed and not reconstructed accurately. However, these curves can still be used for validity assessment of the efforts. The initial part of the effort is always very loud and reconstructed accurately. Therefore, the investigator can still look at the ascending slope at the start of the test. For cough information, I envision that the Hilbert envelopes of the temporal audio data can be attached along with the spirometry curves, which would make any coughs clearly visible. However, in cases where the spirometry curves are of importance, the use of SpiroCall is not recommended. However, I also recognize that a more rigorous evaluation of the spirometry curves is important. This is part of our on-going clinical trials, where we are sending all the curves generated by SpiroCall and SpiroSmart to medical practitioners for quality assessment at Spirometry 360[5].

Figure 5.29: A user using SpiroCall on a feature phone (Sony w580i) with and without a vortex whistle.

5.7 Whistle

The clinical deployments of SpiroSmart have highlighted a number of usability and training challenges.In this section, I discuss a special mouthpiece that has been designed to aid in performing spirometry efforts on SpiroSmart and SpiroCall. This mouthpiece is basically a vortex whistle with no moving parts 5.29. The whistle generates vortices as the user exhales through it [196, 224, 229], changing its resonating pitch in proportion to the flow rate.

The whistle was evaluated along with SpiroCall in a study involving 50 patients. The results show that when compared against two FDA-cleared clinical spirometers, the vortex whistle has a mean error of 6.2% for the four major lung function measures, and for FEV_1% the error is 7.3%. In comparison to without using the whistle, the whistle leads to higher average error in lung function estimation, it performs more consistently for people with lower lung function and produces fewer over-estimations of lung function (*i.e.,* false negatives).

[5]http://spirometry360.org

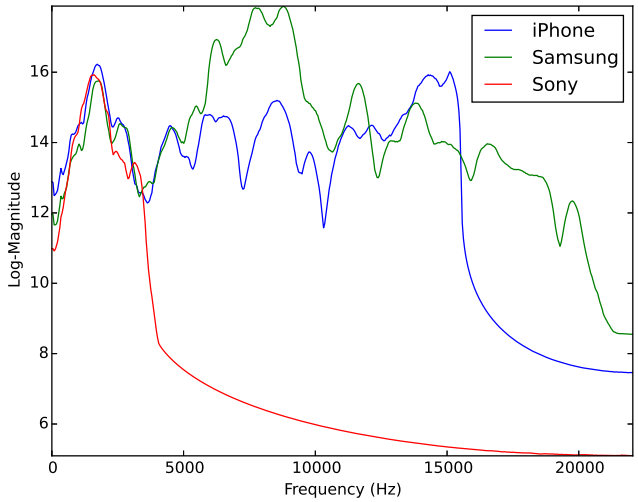

Figure 5.30: Approximate microphone frequency response on three phones used in the evaluation of SpiroCall and vortex whistle.

5.7.1 Motivation

The clinical deployments of SpiroSmart highlighted some very important issues: (1) a patient with severely low lung function might not generate any sound; (2) it is hard to train patients on how to perform spirometry maneuver; and (3) algorithms created from audio collected on a specific smartphone model may not generalize to other models or brands. For example, Figure 5.30 shows the frequency responses of microphones on three different phone models.Moreover, there are other significant usability challenges that are more difficult to mitigate: how a user holds the phone (angle, microphone occlusion, *etc.*), the distance from the user's mouth to the phone, and how wide a user opens their mouth. Some of these challenges were discussed in Section 5.5.2, however it is hard to mitigate them completely; especially when the user is using a feature phone which might not have other sensors that SpiroSmart relies on to counter some of the challenges. The vortex whistle offers several important advantages that can help in mitigating a number of the earlier mentioned challenges: (1) the acoustic properties of the whistle are more consistent than a user's vocal tract and generate audible sounds even at lower flow rates, (2) the whistle removes the effect of distance from the user's mouth, (3) precisely

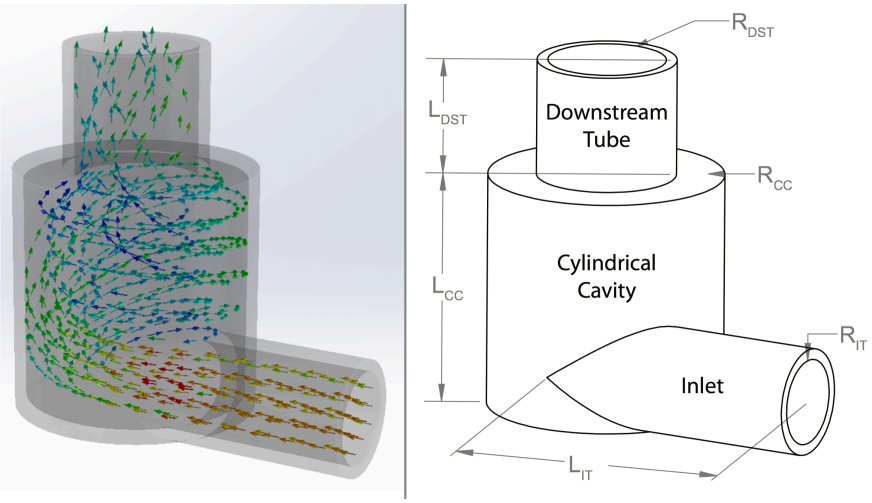

Figure 5.31: ***(Left)*** **3D rendering of the whistle. The arrows show the airflow and the colors denote the velocity with red being faster and blue being slower.** ***(Right)*** **Dimensions (in mm) of the two whistles: Small: R_{CC} = 20, R_{DST} = 11, R_{IT} = 8, L_{IT} = 50, L_{DST} = 24, L_{CC} = 40. Big: R_{CC} = 37.5, R_{DST} = 12.5, R_{IT} = 8, L_{IT} = 74, L_{DST} = 27, L_{CC} = 35.**

controlling mouth shape and phone orientation become less important, and (4) it removes the need for amplitude-based features to compute the lung function. I believe the whistle can complement the traditional no-mouthpiece approach of SpiroSmart to bring reliable spirometry to every mobile phone in the world.

5.7.2 Design

Bernard Vonnegut, in 1954, designed a whistle that changed its pitch in proportion to flow rate and called it a vortex whistle [224]. Later, Watanabe and Sato suggested modifications to vortex whistle construction for use in spirometry efforts [196, 229]. In their study, they used pitch tracking to convert the vortex whistle sound to an estimate of flow rate. Considering pitch tracking is resilient to variations across devices, such as gain and frequency response, the whistle could make SpiroCall independent of distance, channel, and device. The whistle has no moving parts and thus, is as simple as any spirometer mouthpiece—mass-producible for less than 10 cents (US). I decided to test the design proposed in [196] to see if it could be used as a flow-sound transducer instead of the user's vocal tract. However, I found the proposed design unsuitable for spirometry

and modified it based on a pilot study with 15 participants.

The vortex whistle consists of three sections: the *inlet*, the *cylindrical cavity*, and the *downstream tube*. The inlet is a cylindrical pipe that is tangentially connected to the cylindrical cavity on its curved surface. The user blows through this tube. The cylindrical cavity allows the air inside to swirl around the chamber. The downstream tube is attached perpendicular to the cylindrical cavity. When the air enters the cylindrical cavity, it starts rotating along the circumference of the cavity, thereby forming a vortex, and moves toward the downstream tube. The arrows in Figure 5.31 *(Left)* show the result of a simulation of airflow within the whistle. The color of the arrow denotes the simulated velocity of the air. When the air leaves the cylindrical cavity, the vortex becomes unstable and whips around at an angular velocity that is proportional to the rotational velocity of the vortex. This unstable vortex generates sound as it leaves the downstream tube.

The frequency produced by the whistle is affected by several factors, including the dimensions of the whistle [224]:

$$F = \frac{U}{2\pi R_{CC} A} \sin\theta \sqrt{\frac{1}{R_f (L_{DST} + \Delta L_{DST})}}$$

where F is frequency, U is input flow rate, R_{CC} is the radius of the vortex in the cylindrical cavity, A is the cross-sectional area of the inlet, R_f is radius of the air in the downstream tube, L_{DST} is the length of the downstream tube, and ΔL_{DST} is the length of the vortex formed at the outlet of the whistle. The $\sin\theta$ term refers to the angle between the formed vortex and cylindrical plane. This term is difficult to calculate mathematically, necessitating that the quantity be determined through calibration [224]. Typical values range between 0.35 up to 0.95. The design suggested in [224] was modified to ensure that the whistle's response remains linear even at flow rates around 15 L/s. This flow rate is well above the peak flow rate attainable by individuals with height up to 210 cm. I designed two sizes of the whistle (dimensions shown in Figure 5.31, *Right*), as different sizes will have different pitch gradients. Figure 5.32 shows an example input (red, this is the groundtruth recording

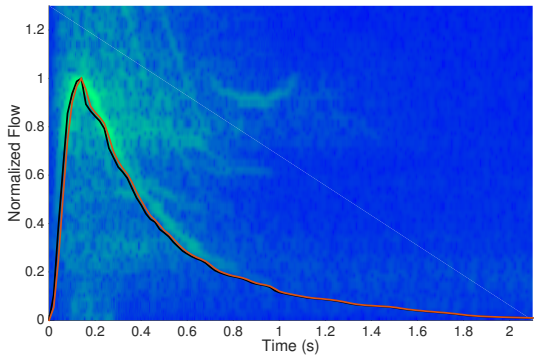

Figure 5.32: An example spirometry effort. It shows the input flow rate (red), output flow rate (black) as modeled by SolidWorks, and the spectrogram received on the phone.

from the clinical device) and output (black) flow rate for the smaller whistle, as modeled by SolidWorks. the plots are overlaid on the corresponding audio spectrogram received on the Samsung Note 3 device. It must be noted that the flow rate plots and spectrograms are not from the same effort because the groundtruth and whistle audio cannot be recorded simultaneously.

I 3D-printed the whistles on a Stratasys BST768 printer using ABS plastic material. The evaluation of both the whistle sizes with 50 participants demonstrated that the bigger whistle performed better because it had a steeper pitch gradient. From a usability standpoint as well, 34 out of 50 participants also preferred using the bigger whistle, because it was easier to handle.

5.7.3 Algorithm

When a vortex whistle is used,the audio processing can be considerably simplified because the whistle pitch changes linearly in response to flow rate. Simple pitch tracking can estimate the flow rate over time. The parameters of this linear relationship (bias and slope) can be calibrated using a few example spirometry efforts. For a particular vortex whistle with set dimensions, these parameters only need to be calibrated once.

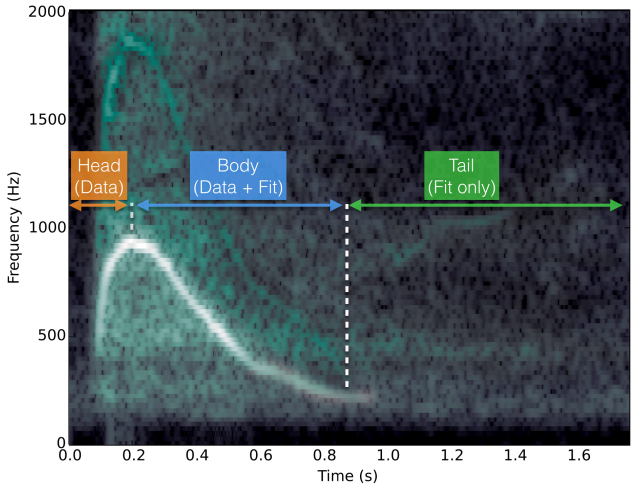

Figure 5.33: The blue and orange regions are associated with the pitch tracked by the algorithm. The green region is extrapolated based on the information in the blue region.

5.7.3.1 Whistle Pitch Extraction

All audio data is resampled to 44.1 kHz to ensure uniformity in the processing across devices with different sampling rates. The first step is to process the spectrogram of the audio recording to track the pitch. The data is then segmented into frame durations of 46 ms with a step size of 3 ms between frames. Next, I find the peak magnitude in the spectrogram (Figure 5.33) and search for the peak frequency within 0.25 seconds. The peak frequency (Figure 5.33, top of the white curve) corresponds to the PEF of the spirometry effort. The pitch is tracked backward and forward in time from that point, stopping when the spectral energy ceases to trend towards lower frequencies. This helps in ignore wheezing at the end of a spirometry effort that may overwhelm the main whistle audio amplitude. For each frame, a quadratic polynomial is fitted to the frequency bin of interest and its two neighbors to achieve sub-bin accuracy in the peak frequency estimates [159]. The pitch tracking stops once the resonance passes below a certain empirically determined pitch threshold, as the whistle mouthpiece does not resonate well at lower flow rates and therefore lower frequencies.

For example, in Figure 5.33, the pitch can be tracked up to 1 s. This means that while one can infer the FEV_1 value from the pitch data, the FVC value needs an extrapolated curve.

5.7.3.1.1 Tail Extrapolation

After the flow achieves its peak value (PEF), the flow rate decays exponentially for a healthy individual and decays faster than exponentially for an individual with obstructive lung impairments. Therefore, while extrapolating the pitch curve, one cannot just use an exponential fit function. I apply a combination of exponential and exponential of exponential fits, so that the system automatically adapts to different types of flow-time curves, including the ones where the flow rate decay faster than exponentially . We fit the following function to the tail end of the flow-time curve:

$$x(t) = (a_0 e^{-b_0 t} + a_1 e^{-b_1 t^2}).a_2 e^{e^{-b_3 t}}$$

The entire descending limb of the tracked pitch is used to fit the extrapolation function. The green area (Tail Fit only) in Figure 5.33 shows the time over which the curve is extrapolated. The blue area (Body, Data+Fit) represents the phase in time during which reliable resonance tracking data is available. However, in order to transition smoothly from resonance-tracking to extrapolation, the algorithm cross-fades from resonance-tracked data to extrapolated data within the blue region. The extrapolation function was evaluated by applying it to the set of groundtruth flow-time curves to ensure it was able to model the tail end of a user's exhalation. Although the extrapolation function worked exceptionally on groundtruth data (mean error = 3.2%), when the extrapolation was evaluated on the audio data received from SpiroCall devices, the FVC estimates had an average error of 15%. Therefore, I decided to estimate the FVC through a regression model, using the above calculated extrapolated curve as a feature in the regression.

5.7.3.1.2 FVC Regression Model

Although the tail extrapolation method did not provide an adequate volume (FVC) measure, it still provided a good, albeit noisy, estimate in most cases. I, therefore, encode the pitch tracking output as a set of regression features. Figure 5.34 shows all the features

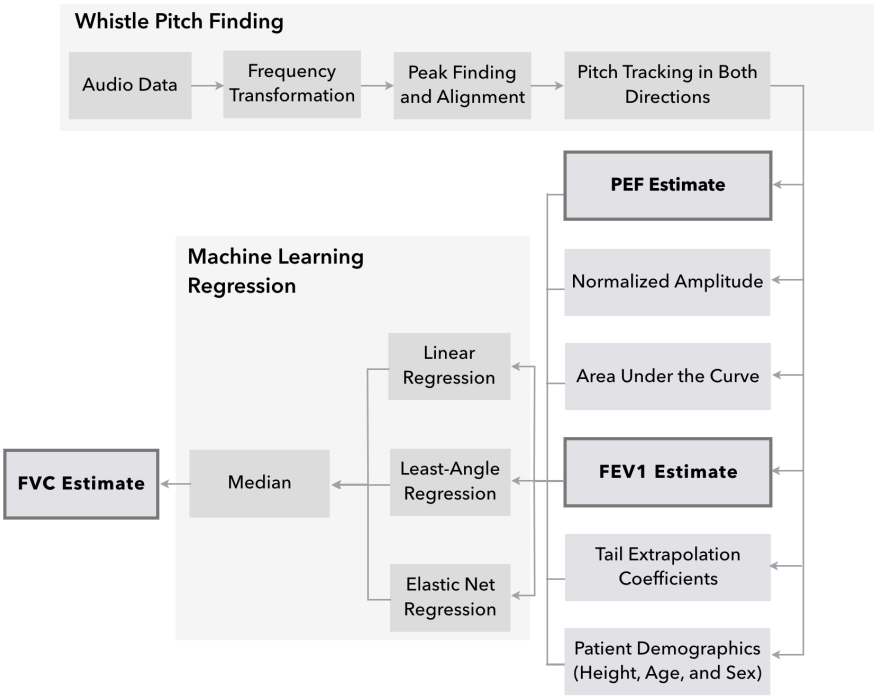

Figure 5.34: Flowchart for lung function estimate using vortex whistle

used in the regressions. The features can be broken down into the three phases of the pitch tracking in Figure 5.33: *Head, Body,* and *Tail*. I use the estimated PEF, *i.e.,* the peak frequency of the tracked pitch, as the representative feature from the *Head* section of the curve. The peak amplitude (normalized) of the overall audio is also used. From the *Body* section of the curve, we use the area under the pitch-tracking curve until the end of the body section, and the area under the curve until the end of 1 sec, *i.e.,* FEV_1 estimate. The next set of features comes from the *Tail* extrapolation. I use the coefficients generated by the tail extrapolation as an encoding of the curve in the *Tail* region of the curve. Specifically, we use a_0, a_1, and a_2 from the tail extrapolation equation as features in the regression. Apart from these features, I also use height, age, and sex as the demographic features. It is common practice in spirometers to record a patient's physical details as this information helps the device in calculating predicted normal lung function for the patient.

The regression algorithm employs an ensemble of three regressions: linear, LARS, and

elastic net regressions. The outputs of all regressions are combined together and the median of their estimates is outputted as the final FVC estimate. I use leave-one-patient-out cross validation in all levels of learning to avoid overfitting.

5.7.4 Evaluation

The vortex whistle was evaluated along with the SpiroCall service in a study with 50 participants. The detail of the study design can be found in Section 5.6.3.

The study used a within-subjects 2×2×3 factorial design. The factors and levels were:

- ***Phone Type:*** *iPhone* and *non-iPhone*. I used two non-iPhone devices: Samsung Note 3 and Sony Ericsson W580i. I used the W580i (feature phone) to evaluate the performance of SpiroCall on an approximately 10-year-old device.
- ***Channel Type:*** *Local recording* and *voice channel recording*. The iPhone was kept consistent in both channels to analyze the performance of SpiroCall if only the channel is changed.
- ***Whistle:*** *No whistle*, *small whistle*, and *big whistle*. The audio data was recorded using two whistles to understand if different participants preferred different sizes or if one size gave results that were more reliable than the other.

All the conditions were counterbalanced and we randomized the order of the whistles. The participants performed at least 15 spirometry efforts (three each for: two clinical spirometers, two whistles, and one without whistle).

5.7.5 Results

5.7.5.1 Comparison of Two Whistle Sizes

I used two sizes of the vortex whistle in our study. Both whistles had slightly different gradient of pitch with respect to the input flow. I performed a two-sample F-test for equal variances on the percentage error for the four lung function measures for both whistle

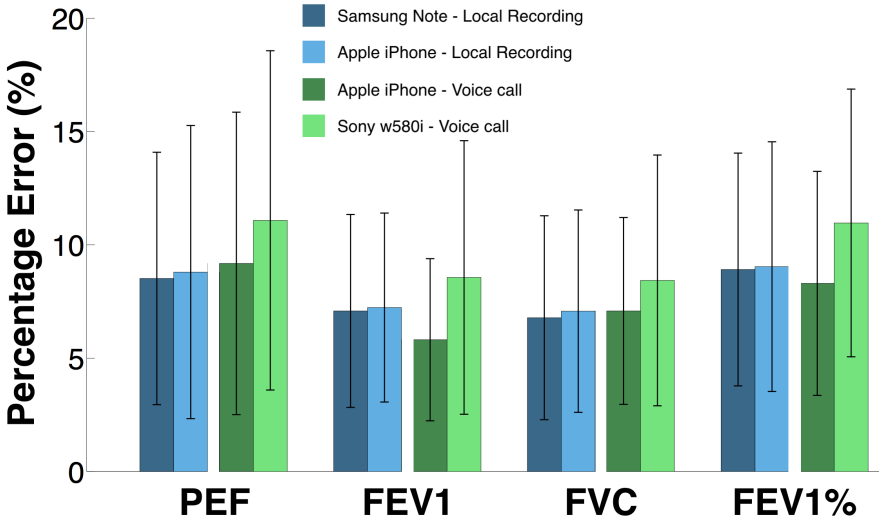

Figure 5.35: Percent error for different lung function measures on different devices with the whistle. The first two devices record the data locally in an app (*i.e.*, SpiroSmart); the next two devices recorded the data over a phone call (*i.e.*, SpiroCall). The error bars show standard deviation.

sizes. I observed a significant effect (p<0.01) of size on FVC and FEV_1% , in favor of the bigger whistle. The percentage difference between the two whistles was 0.24%, 4.22%, 2%, and 2.13% for PEF, FEV1, FVC, and FEV_1% , respectively. Considering the bigger whistle worked significantly better, the analysis only includes the performance of larger whistle from here on.

5.7.5.2 Accuracy of Lung Function Measures

Bar graphs shown in Figure 5.36 display the percentage error of each lung function measure for each device and connection type with a whistle. The Sony Ericsson W580i performed the worst among all the phones. However, the difference was not statistically significant (F-test, p>0.05). Among the lung functions, the error was highest for PEF, but it is worthwhile to note that the variance in PEF was also the highest for the groundtruth spirometers. The most widely used lung function measure, FEV_1% , has less than 8% mean error for three of the four device types.

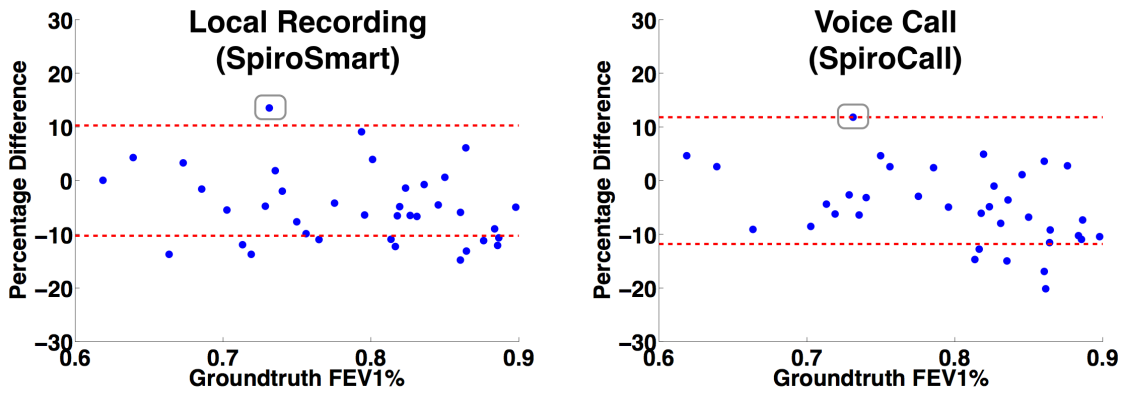

Figure 5.36: Bland-Altman plots of percent error of FEV_1% for local and voice call recordings of whistle *vs.* the values obtained from the clinical spirometer. The false negatives are highlighted insider gray boxed. ±2σ (red dashes) are also shown.

5.7.5.3 Outliers and Patients with Low Lung Function

In order to understand the direction of the bias present in whistle results, Figure 5.36 shows modified Bland-Altman plots of FEV_1% , displaying percentage difference between SpiroCall (with whistle) and the spirometer versus the spirometer measure. From these plots, I show that the whistle mitigates false negatives. The figure highlights the false negatives inside gray boxes. Most of the error for the whistle comes from false positives. When comparing local recordings and voice calls, there is no significant performance difference (F-test, p>0.05). However, the whistle eliminates the bias in the estimate that we saw in case of SpiroCall in Section 5.6.4.3. This means the whistle may be a superior screening tool, especially for patients with very low lung function. This effect of bias is quantified by considering the effects of groundtruth FEV_1% on the percent error through a chi-square test. There was no significant effect of the groundtruth FEV_1% on the accuracy of SpiroCall across devices (p>0.05).

5.7.5.4 Flow Volume Curves

The reader might recall that the Flow *vs.* Volume curves generated from SpiroCall can be noisy and unreliable (Section 5.6.4.4). Figure 5.37 shows example Flow *vs.* Volume curves generated using SpiroCall with and without the vortex whistle. The no-whistle (green)

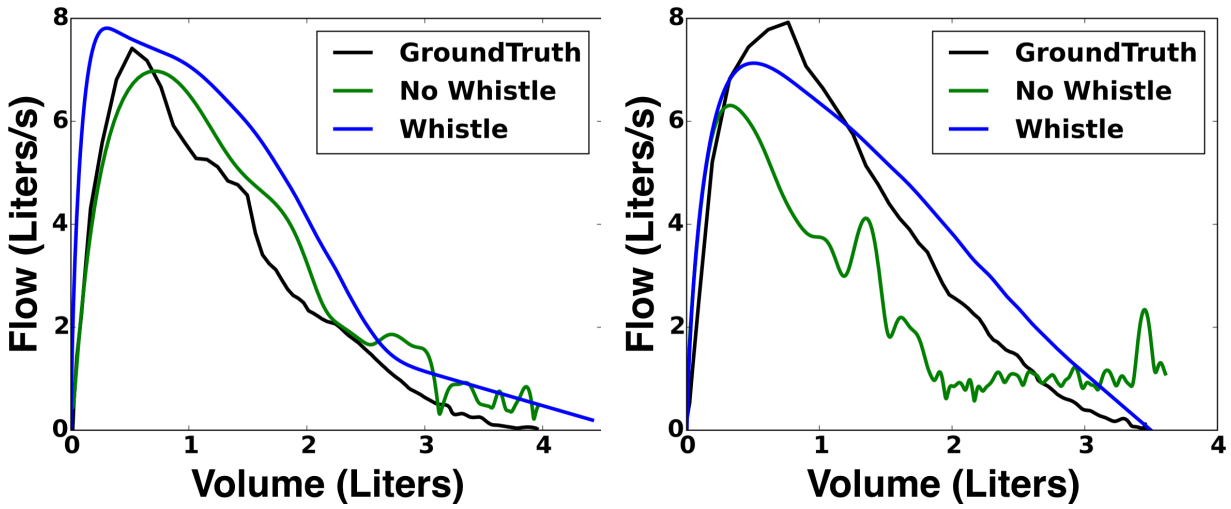

Figure 5.37: Two Flow *vs.* Volume curves generated by whistle (through SpiroCall)

curve in the Figure 5.37 *(Right)* has an inaccurate shape because the latter half of the effort by the patient was very quiet. However, in case of the whistle, the plots are more reliable. In cases where the spirometry curves are of importance, I suggest the use of whistle; especially when using them with SpiroCall (*i.e.,* the call-in service). The whistle generates a direct mapping to the Flow *vs.* Time curve and the final Flow *vs.* Volume curves are usually more accurate.

5.7.6 Evolution of Whistle Design

The reader might recall that the vortex whistles do not generate sound throughout the spirometry effort. As the flow rate decreases at the end of the effort, the vortices formed by the whistle begin to degrade and the emitted sound become undetectable. Therefore in Section 5.7.3.1.1 I discussed a pitch extrapolation algorithm that takes the tail end of the pitch tracking and extrapolates it and then runs multiple regression algorithms to estimate the total lung capacity (*i.e.,* FVC) of the user.

Recently Kaiser *et al.* [105] have improved the design of the vortex whistle presented earlier in this chapter. Figure 5.38 shows the new design. In order to compensate for the weak vortices at the end of a spirometry effort, they added a side whistle to the design.

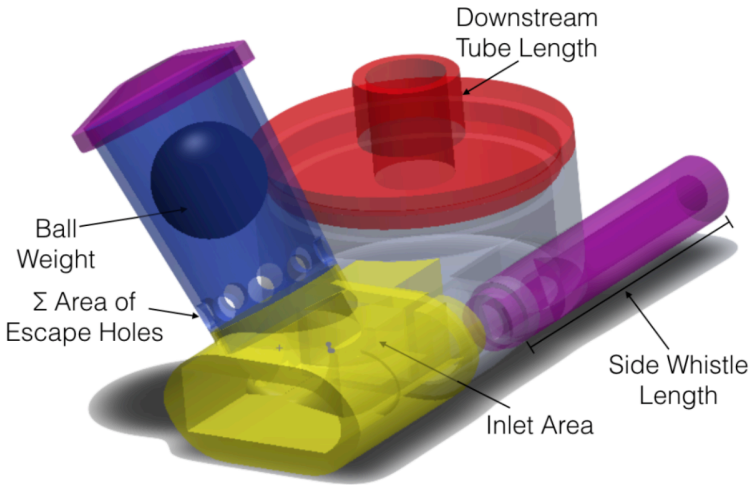

Figure 5.38: Updated design of the vortex whistle by Kaiser *et al.* [105].

Unlike vortex whistle, the aerodynamic whistles generate strong pitches even at lower flow rates. However, this side whistle can interfere with the sound generated by the main vortex whistle. In order to prevent this from happening, Kaiser *et al.* modified the length and diameter of the main chamber. More details of the updated design can be found in [105].

Chapter 6

Recommendations for Future Mobile Devices

The discussion in this chapter is driven by the challenges faced during the development, evaluation, and deployment of various projects that I have discussed in preceding chapters, especially SpiroSmart. The applications that are dependent on generic on-device sensors face extremely varied challenges. It is hard to generalize the solutions across applications, and in each chapter, I have discussed the approaches that worked for that particular application. In this chapter, I discuss some recommendations for future mobile devices. The idea behind these recommendations is to make indirect mobile sensing more potent and uniform across usage scenarios and devices. I have divided the recommendations into three different categories:

1. Generalizing across devices
2. Improving the performance of current applications
3. Adding new capabilities to existing sensors

6.1 Making Applications Generalizable across Devices

One very important challenge faced by many applications using on-device sensors is device fragmentation. There are many different makes, models, and operating systems of mobile devices. This diversity is interesting because although there are many different hardware specifications out there, their capabilities are often very similar. In many cases, this means that a system implemented on a particular operating system and hardware model can very well be implemented on other devices. For example, SpiroSmart uses the microphone, front-facing camera, and inertial sensors . It will be hard to find a smartphone in the market right now without these sensors. So, in theory, SpiroSmart can be implemented on *all* phones. However, the sensors differ just enough between phones that the algorithms will not be directly transferable. For example, SpiroSmart samples the audio data at 32 kHz and the frequency response for different smartphone models starts deviating significantly after 4 kHz. Therefore, the SpiroSmart data collected on Apple iPhone4S and iPhone 5S would not be directly comparable. This can be a huge problem for data-driven machine learning models as mobile devices are upgraded every year. It is almost impossible for researchers to keep up with that pace, especially when it requires collecting data on thousands of patients. Although one can imagine coming up with a transfer function that transforms the data from one mobile device into something that is compatible with a different device, it is not going to be straightforward. I believe that device manufacturers have a huge role to play here. They can provide invaluable support to researchers and developers in making sure that their technology generalizes across a spectrum of devices.

6.1.1 Datasheets

The first and obvious step would be to release datasheets of all sensors on the device. Currently, developers have to pry open a phone using their own technical knowledge or

through online instructions (*e.g.,* iFixit[1]). Then, they must search for sensor model numbers and search for their datasheets. This can sometimes be impossible as the ICs are proprietary or part of a larger SoC. If the manufacturers are able to provide datasheets for different capabilities of their devices as part of their developer program, it will become easier to find that transfer function between different devices.

If such information becomes available for sensors like microphones and cameras, it will become easier for technologies like SpiroSmart to find the transfer function between different devices. Later in this chapter, I will briefly discuss another project, BiliCam, that uses the phone's camera to estimate a newborn's bilirubin levels. Understanding the absorption spectra and the gamma response of the camera will help in spread a BiliCam application to a broad range of devices (all data is currently collected on an Apple iPhone 5S).

6.1.2 API Support

Although most mobile operating system SDKs provide reasonable support to access the on-device sensors, I believe the overall support can be significantly improved. In this section, I focus on two main areas where better support would be useful:

1. Sensor diagnostics
2. Full access to specific-purpose sensors

6.1.2.1 Sensor Diagnostics

Modern operating systems often perform quick diagnostics of their hardware and inform the user if something is deemed wrong. For example, if the phone overheats, it often gives an appropriate notification to the user and turns off for some time to cool down. Similarly, some new phones are rumored to detect if there is possible water damage to some of the

[1] www.ifixit.com

hardware and then notify the user so that they can fix it before the hardware is damaged[2]. Similar support for on-device sensors and giving developers access to detailed diagnostics reports will prove extremely useful.

With the advent of phone-based medical, the onus on these on-device sensors is so high that it is important to be completely sure that the sensors are not broken or malfunctioning. For example, microphones can get slightly blocked due to lint or dust deposit. This can reduce the microphone's response to some frequencies. If the phone allowed applications to run a quick diagnostic on the phone's microphone, applications could check if the frequency response of the sensors has changed significantly. Similarly, a device's CMOS sensor can become damaged or the light hitting the sensor might start leaking due to some hardware deformity; or the camera lens might just have smudges and fingerprints on the lens. It will be very important for a camera-based health application to be certain that the camera is working as expected before the user uses the device for an important health checkup. Performing deep and frequent diagnostics on phones and allowing applications to run them as well will make devices significantly more reliable and make a much stronger case to get regulatory approvals. In many cases it will be difficult to assess the performance of the sensor without a calibrated actuator (*e.g.,* BiliCam's color calibration card [53]) and I believe having such actuators would slowly become essential to the success of mobile health-sensing applications.

6.1.2.2 Full Access to Special-Purpose Sensors

Special-purpose sensors refer to sensor that are added to the device for a very specific purpose. Examples include noise-cancellation microphones and proximity sensors. These sensors often only return a high-level value to the operating system that is then exposed to the developer. For example, in the Android OS, the developer has the option of getting the noise-cancelled audio using the "NoiseSuppressor" class[3]. Such constructs are

[2]http://appleinsider.com/articles/16/07/20/liquid-ingress-warning-in-ios-10-beta-protects-iphone-against-water-damage-

[3]https://developer.android.com/reference/android/media/audiofx/NoiseSuppressor.html"

extremely powerful and useful for third-party applications. However, these do not give access to the raw signal captured by the noise-cancellation microphones. Another example is the camera flash. Many devices have multiple flash sources with their cameras, but the SDK to control them is usually very sparse. Developers seldom have the option to control which flash they want to turn on and for how long.

These are all isolated examples that might be fixed soon; however, I aim to highlight a bigger concern here. When these specific sensors are added, manufacturers often do not foresee innovative uses for these sensors outside of their original intent. Providing these developers complete access to these sensors will, in the end, be beneficial to end users.

6.2 Improving Performance of Current Sensors and Applications

There are a number of improvements and modifications that can benefit all mobile device sensors. For example, many researchers and developers desire higher sampling rates for sensors like microphones, cameras, and inertial sensors that would allow them to better capture high frequency information. In fact, it is hard to think of a sensor that would not benefit from an increased sampling rate.

Similarly, having a more battery-efficient sensing infrastructure would be very useful. Computation can be offloaded to servers through wireless communication, but that requires stable Internet access. Transmitting data also raises concerns about data privacy, which is particularly important for protected health information. Performing computations on the devices themselves alleviates these concerns. Techniques like deep convolutional and recurrent neural networks make a very strong case for continuous, high-granularity data from sensors. If sensors need to generate high quantities of data, it is also pertinent for them to sample continuously and more efficiently. Using co-processors for some of the sensors was probably the first step in this direction [138, 177, 209], and many of today's smartphones include co-processors for

speech recognition or IMU analysis. One of the main reasons for this change is step counting and other fitness applications. These applications motivated continuously accelerometer sampling. Perhaps, that is why some of the initial commercial co-processors were called motion co-processors [4]. The next class of sensors to be off-loaded to these co-processors was microphones [72, 141]. In this case, the strongest push probably came from voice assistants on mobile devices (*e.g.,* Google Now, Apple Siri, *etc.*). The performance gain in using these co-processors is so strong that researchers have recently suggested ways to automatically divide application code to leverage these co-processors without any programming effort from the application developer [205]. Researchers are now also suggesting similar techniques for cameras [136] where the data is processed by an analog signal processing layer that is part of the sensor co-processor. There have been similar proposals for microphones as well and modern smartphones use them for their voice assistants. The reason this is especially attractive for microphones and cameras is because these two sensors generate data at a significant rate, so it becomes important to detect the "region of interest" as soon as possible so that the device doesn't run out of memory.

6.3 Adding New Capabilities

The topic of adding new capabilities to existing sensors is again quite broad and I would like to constrain it to one central idea: go beyond the human capabilities.

Devices like microphones and cameras can be considered as ways to document human experience. They are optimized in a way that mimic human capabilities: the microphone has a cut off frequency around 19 kHz and the camera is sensitive to the visible spectrum. As we start thinking of these devices more as sensors, we can think about going beyond human capabilities. For example, many researchers have explored ways of leveraging the tiny frequency band between 18.5 kHz and 20 kHz [12, 36, 82] where microphones are

[4] https://en.wikipedia.org/wiki/Apple_motion_coprocessors

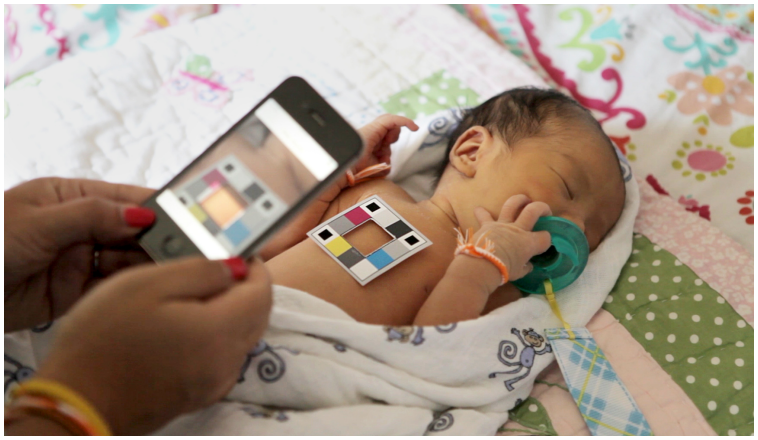

Figure 6.1: Using BiliCam, parents or medical practitioners can monitor a newborn's jaundice with their phones.

sensitive, but most human adults are not. By exploiting the range outside human sensitivity, one can develop something that does not interfere with the user's senses and environment.

Avoiding interference is not the only reason for going outside the range of human sensitivity. There are many phenomena around us that we are unable to sense. It would be great to have a tool that helps us observe phenomena occurring outside our sensitivity range instead of just helping us document what we can already observe. Cameras are a great example for such an approach. They are already very powerful, and the fact that we have a device that can document human perception with such precision is very handy. However, there is a chance to go further here. In the rest of this section, I will first briefly discuss *BiliCam*, a system that uses a mobile device's camera to very reliably estimate bilirubin levels in newborns. It serves as an example of the utility of an already very powerful sensor. Then, I will discuss *HyperCam*, a system that demonstrates the power and challenges associated with going outside the range of human sensitivity for cameras, *i.e.,* hyperspectral imaging.

6.3.1 BiliCam

BiliCam [53] is another example of a built-in phone sensor-based medical application (Figure 6.1). It uses the phone's embedded camera to monitor newborn jaundice.

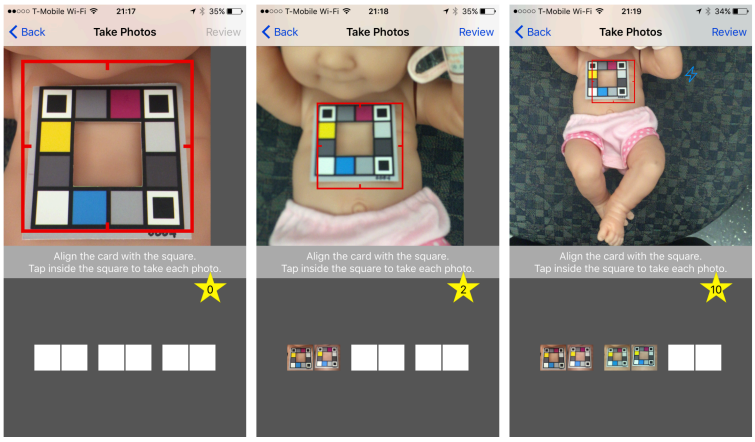

Figure 6.2: BiliCam UI: The user takes photos at three distances between the phone and the baby's chest. The UI guides the user with a view-finder that resizes to help the user change the distance.

Jaundice is defined as the yellow discoloration of the skin. It is caused by excess of a chemical byproduct of recycling of old blood cells: bilirubin. An estimated 84% newborns develop jaundice in their first week. A moderate level of bilirubin is normal. Cases where the bilirubin reaches extreme levels can cause devastating and irreversible lifelong conditions if not detected and treated. The deployment of BiliCam at 8 medical sites across United States and with data from over 500 babies has shown that BiliCam, an app that simply uses the phone's camera, even works better at screening babies than the currently FDA cleared screening tool: Transcutaenous Bilirubinometer (TcB).

Figure 6.2 shows a screenshot of the custom iPhone application used for data collection. For each sample, the app records 6 photos: an image with the flash turned on and an image with the flash turned off for each of the 3 different distances between the baby's sternum and the phone. The technician first placed a color calibration card on the baby's chest and then aligned the card in the bounding box shown in the app's viewfinder (Figure 6.2). Once the card was correctly aligned, the app automatically captured the photo an then auto-advanced. The color calibration card and the view-finder in the interface highlight a very important point associated with the generic sensors-based medical applications: these sensors are inherently unconstrained, but if they are to be used for precise measurements, it is important to impose constraints. The distance measurement in SpiroSmart is another

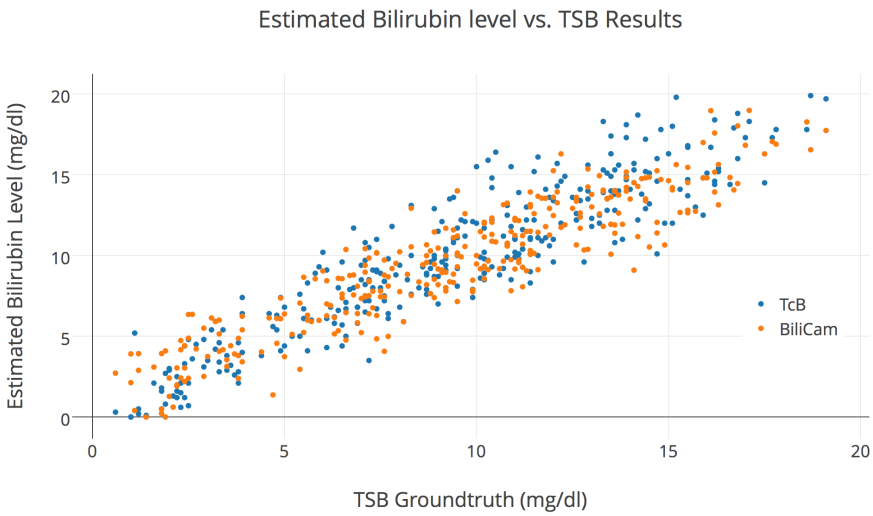

Figure 6.3: A comparison between predicted bilirubin levels and TSB.

example of the kind of constraints that would be important in medical applications. The details of the algorithm can be found in [53].

Figure 6.3 shows a comparison between the predicted bilirubin levels from BiliCam and TcB, against the ground truth values received from the TSB blood test. BiliCam has a rank-order correlation of 0.921, whereas TcB has a rank-order correlation of 0.915. There is no significant effect of the baby's ethnicity on the error in estimation. Across the entire dataset, 12 babies had a high bilirubin (>15 mg/dl). TcB gave a lower result for 3 of these babies (25%), whereas BiliCam gave a lower bilirubin estimate for 1 of these 12 babies (8.34%).

The current study shows that BiliCam, by just using the built-in camera of a smartphone, can be at least as good as an FDA-cleared medical device in measuring newborn bilirubin levels. However, there are a number of challenges that must be overcome before BiliCam can be used by the masses. The lack of generalizability across devices is the biggest hurdle. It is yet to be tested if the results from an Apple iPhone 5S will transfer easily to other phone models. The phone's camera's absorption spectra, gamma response, and flash spectra all influence the data, which highlights the need for developers to have a more intimate access to these built-in sensors. I hope that efforts such as SpiroSmart and BiliCam will help convince manufacturers to provide better support for these sensors. Another important

aspect that should be highlighted is that the phenomenon detected in BiliCam was visible to the human eye, but the untrained human eye is unable to accurately estimate the bilirubin value. When a sensor that mimics the human eye was combined with the memory and learnability of a computer, it outperformed humans. In other applications, however, the phenomenon is not visible to the human eye. This is where thermal and hyperspectral imaging are useful.

6.3.2 Going Beyond the Visible Spectrum: HyperCam

The human eye and RGB cameras divide visible light into three bands of color (although with slightly different spectral responses). When two materials appear similar to the human eye, it only means that they share similar spectral properties when analyzed by the human trichromatic color vision system; those materials can still have very different spectral properties in some other part of the spectrum. Information in other color bands throughout and beyond the visible spectrum remains hidden the human eye and RGB cameras. Hyperspectral imaging provides more dimensions that could enhance the utility of cameras as a general-purpose sensor. In fact, hyperspectral imaging is already being used in the food and agriculture industries [144, 145, 220], astronomy [111], geological mapping [111], and surveillance [208] because of its ability to expose features of an object that are difficult or impossible to detect with the human eye. Although hyperspectral imaging can "expose secrets of the universe" [44], detect hidden diseases [135], predict ripeness and probably even sweetness of fruits [144, 145, 220], the use of hyperspectral imaging has been highly fragmented.

This section discusses HyperCam (Figure 6.4), a low-cost hardware implementation of a hyperspectral camera using time-multiplexed illumination and a software algorithm that makes it substantially easier for users to uncover salient pieces of information for a particular scene. Since multispectral imaging inherently provides an expanded view of the spectrum, it can lead to a far greater number of spectral images for each scene. This much information can be unwieldy and impedes the rest of the design process (*i.e.,* the computer vision algorithm that processes the image data). Users can benefit from

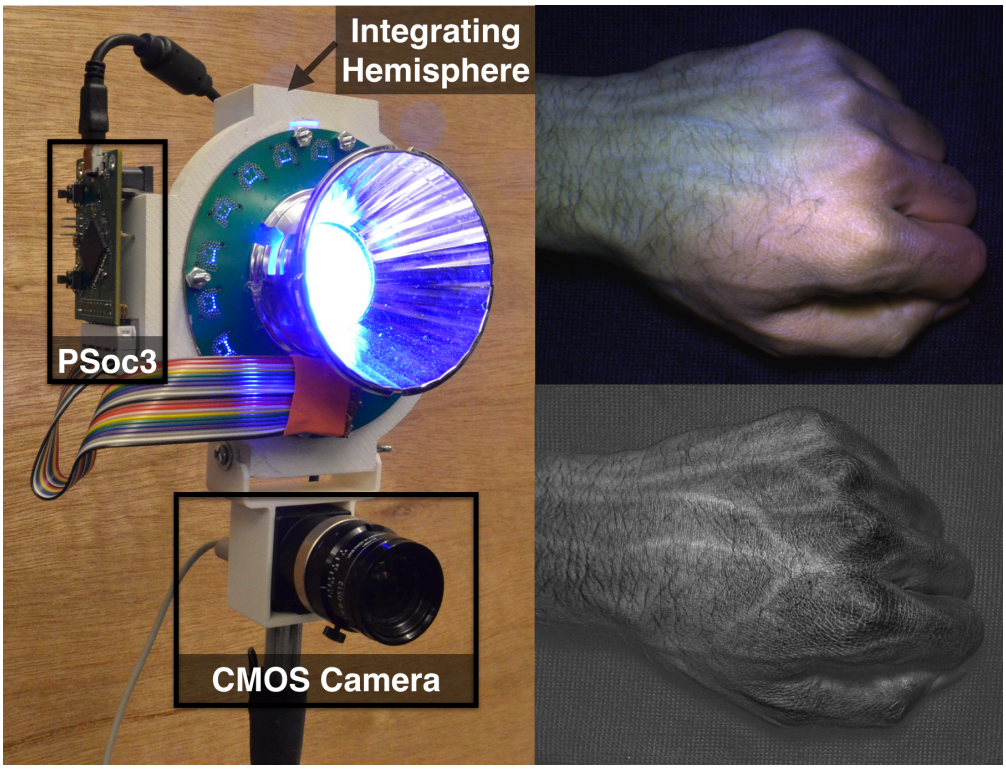

Figure 6.4: *(Left)* HyperCam Hardware. *(Top-Right)* RGB image of a user's hand. *(Bottom-Right)* The user's veins and skin texture are emphasized in the HyperCam-generated image.

HyperCam because it automatically highlights salient aspects of a scene to provide a starting point for further exploration. Such an approach might become important for all sensors that extend beyond human perception. Vision is perhaps presents a more curious case because humans can still see most of the objects through the camera, but not the underlying phenomenon.

HyperCam first captures the relationships between different wavelengths of light by using a set of transformation techniques, including PCA and normalized linear combinations of the collected sequence of spectral images. The system then automatically analyzes this new set of images to determine what results would be "interesting" to the user. "Interesting" images in this context are defined as images that show information not visible in RGB alone. The system automatically adapts to each scene and requires minimal input from the user. Depending on whether the user is looking for spatial or temporal changes in a scene, HyperCam modifies its output accordingly. In the end, HyperCam provides the user with an optimal set of images that it believes best captures the variability in the scene. For example, when a user places his or her hand in front of the camera, the system produces an image like Figure 6.4, *(Bottom-Right)*. This image accentuates the user's skin texture and vein patterns, which are not seen as prominently in the RGB image.

6.3.2.1 Applications of Hyperspectral Imaging

Hyperspectral imaging has largely been used for remote sensing, surveillance, and industrial applications, where spectral signatures are used to differentiate between materials. Remote sensing is one the most popular uses of hyperspectral imaging. GLIMS uses satellite imagery to analyze glaciers' extent and changes [16]. Researchers have also used hyperspectral data from satellites to predict landslides [208]. They monitor vegetation and land cover from hyperspectral images and then make models to robustly predict the landslide susceptibility of an area. In areas where vegetation covers less than 40% of the area, soil and rocks make it difficult for traditional imaging techniques to accurately predict vegetation cover [207]. Hyperspectral imaging helps in such situations

because green vegetation has a distinct reflectance spectrum in the visible and near-infrared (NIR) regions.

Another very popular area for the use of hyperspectral imaging is non-invasive analysis of works of art [18]. It can be used to infer the painting techniques of different artists and detect forgeries. Relatively recently, NIR imaging and spectroscopy has also gained popularity for applications in physiological sensing. For example, biomedical researchers are working on imaging approaches to accurately measure blood glucose levels [89, 117].

6.3.2.2 HyperCam's Hardware Design

For most multispectral imaging applications, it is desirable to have a high spatial resolution hyperspectral camera that can take 2D images. There are surprisingly few off-the-shelf hyperspectral cameras; the ones that exist are quite expensive (upwards of $5,000 USD) and usually do not have a high spatial resolution. Thus, we decided to build our own high resolution, time-multiplexed multispectral imaging system.

HyperCam (Figure 6.4) is a general-purpose hyperspectral imaging system for visible and NIR wavelengths. We chose to use the PointGrey Flea3 FL3-U3-13Y3M CMOS camera. It is sensitive from 350 nm to 1080 nm, with peak quantum efficiency at 560 nm. This camera has a frame-rate of 150 FPS and a maximum resolution of 1280×1024. We use 17 different spectral bands that are created using narrow band LEDs. The wavelengths for these LEDs vary from 450 to 990 nm. These are off-the-shelf LEDs, empirically selected to cover the camera's sensitivity range. Figure 6.5 shows the power spectral distribution for the selected LEDs. Ideally, each response would be narrow in order to capture the reflectance of only a single band of color. However, there is a tradeoff between LED response width, sampling density, and light efficiency. Additionally, the LEDs should have the same intensities, but we were limited by what is available off-the-shelf for both properties. Thus, there is a need to calibrate the intensities of all the LEDs. We will discuss the calibration process in the next section. The LEDs are arranged in a circle (Ø = 5 cm) around the camera lens (Figure 6.6, *Left*). We time-multiplex the spectral

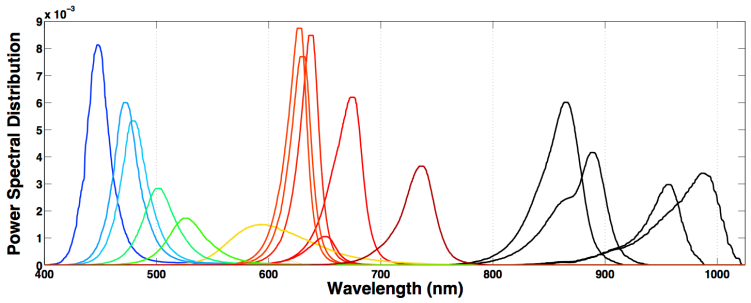

Figure 6.5: Power spectral distribution of LEDs on HyperCam. The color of the curve corresponds to the color of the wavelength. Black color denotes the infrared wavelengths.

bands, and we can capture between 9-150 FPS depending on the number of wavelengths used. HyperCam currently has 17 different bands of spectrum, so if an application needs all 17 bands, then the effective frame-rate for the camera is $150 \div 17 \approx 9$ frames per second.

Given the ring arrangement, each LED has a different lighting direction and path. This causes non-uniform glares and shadows for different wavelengths. We compensate for this by using an integrating hemisphere to diffuse the light and minimize directional non-uniformity (Figure 6.6, *Right*). The light from an LED strikes the integrating hemisphere and then reflects out of the light box through an opening at the center. The integrating hemisphere causes a large number of reflections inside the light box, which is what diffuses the light, but there is some light absorbed at each reflection. Although this leads to reduced light intensity, we found that the final light intensities were satisfactory for most applications. The PointGrey Flea3 camera has a GPIO interface for power, triggering, serial I/O, pulse width modulation, and strobing. We control the camera through the GPIO interface using a PSoC3: CY8C38 chipset. We program state-machines in the on-board EEPROM to enable fast switching between LEDs.

6.3.2.3 HyperCam's Software Design

Time multiplexed hyperspectral imaging can be viewed as generating a 3-dimensional data structure of a scene. The first two dimensions contain the spatial data and the third dimension contains the spectral information. Therefore, the minimum number of

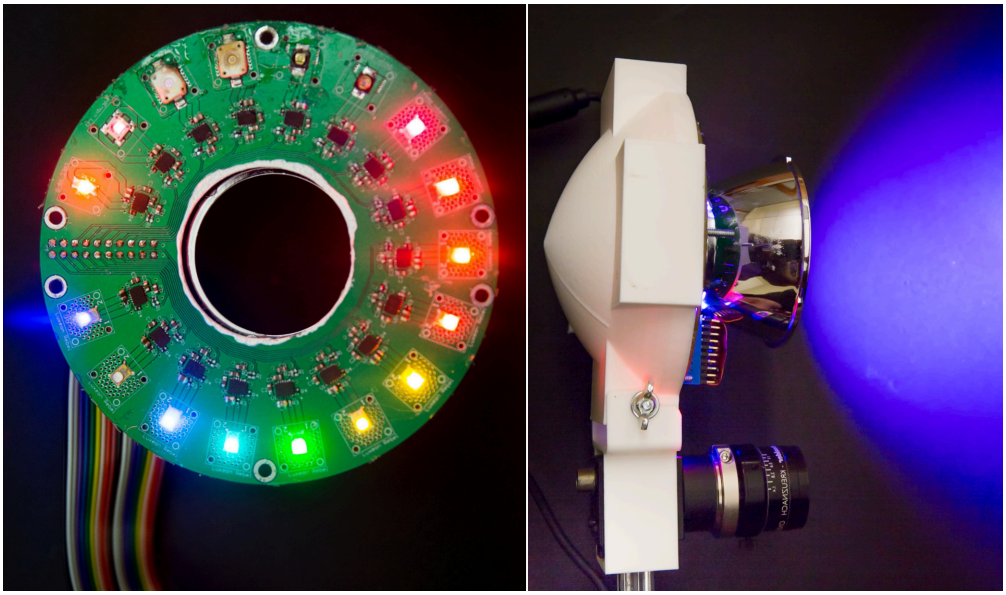

Figure 6.6: *(Left).* 17 LEDs of different wavelengths for HyperCam's spectra. *(Right).* The LEDs shine inside the integrating sphere (white enclosure) and the light comes out of the opening in the center.

luminance images an imaging system generates per frame is the same as the number of wavelengths covered by a camera.

In a number of hyperspectral imaging efforts, the goal of the research is targeted towards a specific application, in which the researchers first identify the wavelengths suited for that particular application *a priori*. After that, a common way of reducing the number of images is using dimensionality reduction techniques such as Principal Component Analysis (PCA) and clustering. The results from this process are then often fed into a machine learning classifier for that particular domain [18, 75].

In contrast, HyperCam is designed as a general-purpose system that can be used for a number of applications. HyperCam captures 17 wavelengths, which means in absence of any further processing, the user would need to sift through at least 17 images per capture. These wavelengths dictated by the selection of LEDs on the device, so the user can switch in and out different wavelengths (by changing the LEDs) as they gain more insight into their problem. In order to maximize the amount of information in a single image,

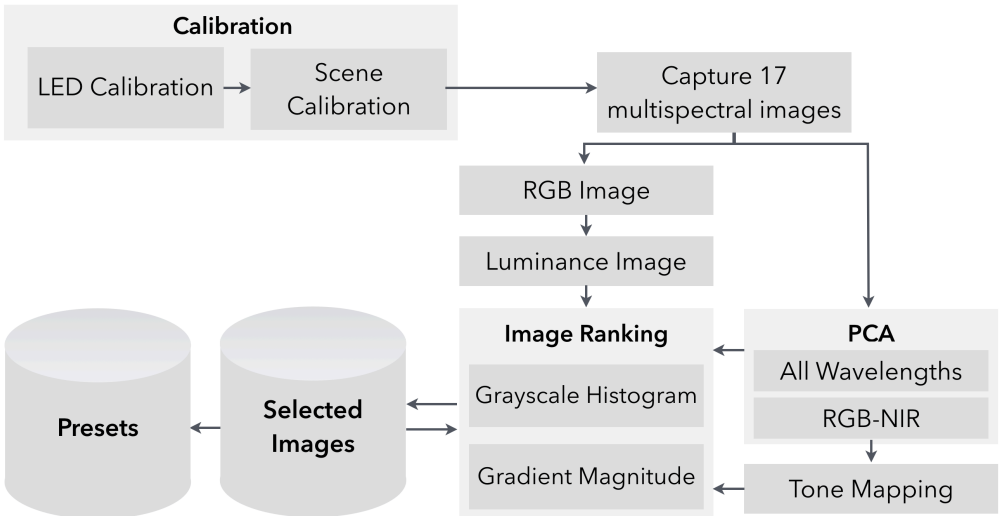

Figure 6.7: HyperCam flowchart.

HyperCam first produces a number of images that are combinations of different wavelength responses. It then ranks them according to a heuristic based on the image histogram and gradient magnitude. The final selected images are returned in the form of images, videos, and/or presets. These presets can later be applied to other scenes and can save the user from computing the desirable combination of wavelengths again. Figure 6.7 shows the step-by-step process used for HyperCam. Figure 6.9 shows the output of HyperCam when a user puts their hands in front of the camera. This output has been shown as useful in authenticating and identifying users in [74]. Figure 6.8 shows the output of HyperCam when some fruits were put in front of the camera. It shows that a multispectral camera can be used to estimate fruit's ripeness and quality. It can prove to be a very useful capability on a consumer mobile device.

6.3.2.4 Implications

Although HyperCam is almost as big as a multispectral camera with a filter wheel (similar to the one used in [18]), the use of time-multiplexed illumination can lead to a substantially smaller device. The device simply needs multiple illumination sources and the CMOS sensor itself. Every smartphone comes with at least one light source, and it is

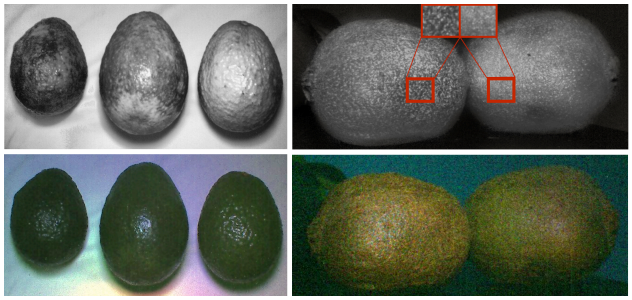

Figure 6.8: *(Top Row)* Automatically generated tone-mapped image. The fruits are arranged in decreasing order of ripeness, *i.e.,* the left ones are riper. *(Left)* Avocado. *(Right)* Kiwi. Avocados display marked difference in luminance with change in ripeness. The call-outs for kiwi show patches from the fruits and put them side-by-side for better visual understanding. The patch from the left fruit is less bright than the right one. Less bright means riper fruit. *(Bottom Row)* RGB image, Note: The images are cropped to provide a closer perspective.

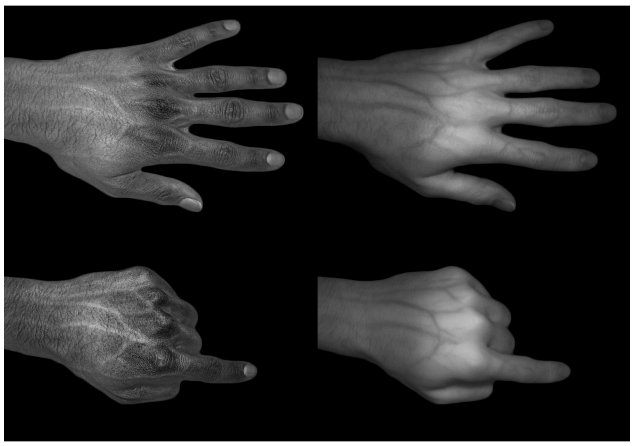

Figure 6.9: Hand images captured by HyperCam that accentuate either hand texture *(Left)* or venous structure *(Right)*

not hard to envision one with multiple such sources. In fact, apart from iPhone's multiple flash LEDs, Eigen Imaging Inc.[5] sells a smartphone accessory that adds such LEDs for NIR imaging and fluorescence detection. The goal of this section was to demonstrate the possibility of going beyond the capabilities of human perception on mobile devices. It shows that sensors are already very close in attaining some of these capabilities. If and when it actually happens, it will raise a number of usability challenges. Since these phenomena will be at the boundary of the user's perception, the onus will be on application researchers and designers to reveal the desired data.

[5]http://www.eigenimaging.com

Chapter 7

Discussion and Conclusion

7.1 Discussion

In the previous chapter, I reflected on my experience in development and deployment of various projects to discuss the evolution of the mobile devices in future. The chapter made a number of recommendations that can improve the state of indirect sensing using generic sensors in future. However, indirect sensing might not always be feasible and a direct sensing approach or even a dedicated hardware might be more desirable. There are some obvious trade-offs between direct and indirect sensing and therefore, it is important to decide between the two. The decision hinges on multiple factors, including the degree of change to the software and hardware, cost of the solution, *etc.* This section explores the trade-off between direct and indirect sensing in more detail and investigates various factors that might affect the final decision.

7.1.1 Degree of Change

In order to enable a new application or improve the performance of an existing application, the developer might make a change to the software, hardware, or both. Within each of these categories, the degree of change might vary as well. This dissertation

mostly explores application-level software changes and basic hardware modifications. However, many applications might involve much more involved modifications to the software and/or hardware.

1. **Application-level Software.** This refers to the cases where the sensors provided by the device's API are used as is. The modifications are purely in the application itself as it uses the observations from the sensor(s) to infer some new information. Most of the applications discussed in this thesis belong to this category. Considering the implementation resides inside an application, the deployment and distribution burden is extremely low.
2. **Background or OS Service.** This refers to the cases where the sensor data collection and the inference runs in the background on the device. This approach is used in cases where the capability is not limited to a particular application. One example of such a system is GripSense. GripSense senses the hand grip using the on-device gyroscope. This capability can be useful in many applications, including the operating system itself. Therefore, it needs to run in the background. Hence, the computing and power efficiency of such algorithms become very important factors.
3. **Firmware Modification.** In these cases, the firmware of the device is modified to enable a new functionality. In such cases, the hardware has a certain capability but, for variety of reasons, it is not enabled in the operating system. For example, Holz *et al.* developed an authentication system for mobile devices by modifying the firmware of the phone's touchscreen. In such cases, it is important to evaluate the practicality of the firmware change. In some cases, if there is a market-friendly application, the manufacturer might prefer adding a dedicated hardware.
4. **Minor Hardware Modification.** An example of such an approach would be the addition of a contact microphone in SurfaceLink. In some cases, making such small hardware changes can be very powerful but they also increase deployment burden.
5. **Interfacing Existing Hardwares.** For example, Bluetooth-connected blood pressure monitors that share the data with the user's smartphone. The sensing and inference component of these blood pressure monitors is same as the traditional monitors, and

the smartphones are also unmodified. The manufacturer adds the interface between these two devices in order to make it more user-friendly and enable new features, such as longitudinal health tracking.

6. **Custom New Hardware.** For example, HyperCam. For some problems the current hardware and form-factor might be ill-suited, then it becomes essential to make completely new hardware and inform the design of future consumer devices.

The optimum approach from the above-mentioned set will always be problem-dependent and will be affected by a number of factors. Next section discusses some of these factors and how they affect the decision.

7.1.2 Factors Affecting the Selection of Sensing Approach

Coming up with an exhaustive list of factors governing the selection of an approach for a sensing solution is impossible. Different problems will have their own complications and niches. In this section I highlight a subset of the factors that a developer must evaluate before selecting an implementation approach. I have divided these factors into three categories:

1. **Performance Requirement.** Different applications will have different performance requirements and at some point, for many applications, the performance will reach a point of diminishing returns. For example, while typing on touchscreen keyboards is hard, still it is difficult to imagine a situation where the user would find it useful to add a new, dedicated sensor that helps in improving text entry. The keyboard is *good enough*. However, not all applications are similar and the gross accuracy number might not always be the best performance metric. For instance, some applications might need a solution that is sensitive to false negatives (*e.g.,* most health applications) or some other application might put premium on delay in inference (*e.g.,* most gesture recognition systems). As shown in case of SpiroSmart and GripSense, while the on-device sensors are reasonably accurate, there is still room for improvement in terms of false negatives and delay,

respectively. Therefore, it is not difficult to imagine a situation where the performance requirements just cannot be satisfied by the on-device sensors and the developer needs to implement something more dedicated and direct.

2. **Usage Environment.** Usage environment is perhaps as important a factor as performance requirement here. Depending on the problem, the system might be used in very *diverse settings*. For example, a 24 hour cough monitor would be to work well in many different conditions; however, a spirometer does not need to work in noisy environments and it just needs to alert the user if the environment is noisy. Indirect solutions are typically more susceptible to noise than dedicated solutions. The developers need to make sure that they evaluate the diversity of the typical usage environments and make sure than their chosen implementation is well-suited to those environments. Another very important property of the usage environment is *unpredictability*. One example of unpredictability in usage environment was highlighted with SpiroSmart. When using SpiroSmart, the user can hold the phone at any distance and orientation. In contrast, a clinical spirometer is always used with a mouthpiece. Therefore, SpiroSmart needs to be resistant to the unpredictability in the user's posture. At the very least it needs to alert the user if their posture is incorrect. I use the phone's front-facing camera to detect the user's posture and the device distance. However similar solution might not be possible for other problems and the developer needs to keep in mind such trade-offs and evaluate the benefit and pitfalls of using an indirect sensing approach.
3. **Economics and Logistics.** Most sensors that make it to the final consumer product make economical and logistical sense to the manufacturer. A general-purpose consumer device, such as a smartphone, only has hardware capabilities that are frequently used by a large proportion of the user base. For example, a dedicated sensor that solves a problem that only affects a fraction of the world's population will always be a custom installation. Even if the problem is a worldwide menace (*e.g.,* HIV AIDS), the proposed sensor would perhaps never become an on-device sensor. Such problems are best suited for a solution that involves least modification to a consumer device. In contrast, let us take the case of the capability to sense the

user's applied pressure on a touchscreen. If the manufacturer believes that such a feature can significantly improve the user experience, they can easily add pressure sensitivity to their hardware instead of taking the indirect approach taken by UnderPressure. Here I am not suggesting that technologies like UnderPressure are useless. Such technologies offer at least two advantages: (1) they bring similar capability to older devices, and perhaps more importantly, (2) they serve as demonstration of the capability and inform the design of the future devices that might as well use dedicated sensors to enable similar capabilities.

As mentioned earlier, this discussion does not aim to provide a exhaustive list of factors and considerations. It provides a set of recommendations and hopes to convince the reader that there is no panacea here. Different problems will benefit from different perspectives and there are no clear winner between indirect and dedicated sensing.

7.2 Conclusion

In this dissertation, I discuss:

1. How to extend the capabilities of on-device sensors,
2. Some of the challenges developers and users may face while using these generic sensors,
3. Discuss how some of the challenges can be countered by combining multiple generic sensors, and
4. Provide some direction as to how these sensors should evolve in future.

This dissertation discusses opportunities and challenges from the domains of health sensing and interaction techniques. I provide several examples of how various on-device sensors (*e.g.,* camera, microphone, accelerometer, gyroscope, touchscreen) can be

extended beyond their traditional capabilities. These sensors typically mimic human capabilities and are severely under-utilized as they are merely used to document the human experiences. This dissertation shows that these sensors can be extended into many more "software sensors" that approximate the capabilities of other sensors that one might not find on a consumer mobile device. The generic nature of these sensors means that they sense many things and can be extremely noisy. In this dissertation, I suggest that combining information from multiple sources can reduce some of the sensor noise. These multiple sources might be: (1) different kinds of sensors on the same device, (2) similar sensors on multiple devices, or (3) some external device/object. I use examples of different works to demonstrate the power of such approaches. I also discuss the challenges that are encountered when such sensors are deployed in noisy real-world settings and the various trade-offs between seamlessness and accuracy. Building on my experience of developing and deploying various sensing systems, I make suggestions for the evolution of these generic sensors.

I first discuss how these sensors can be used to make the devices more context-aware. Unlike humans, these devices do not typically use their sensors to adapt to their surroundings. However, adaptation becomes important as devices are increasingly used in changing situations and environments. I extend the capabilities of various sensors on the phone to make the device adapt to change in user's posture and motion. Through a number of controlled evaluations, I demonstrate the feasibility and benefits of such systems. I discuss a motion adaptive keyboard, *WalkType*, that leverages the on-device accelerometer to compensate for vibrations and extraneous movements caused by walking. Then I discuss *GripSense*, a system that leverages the touchscreen and gyroscope to infer the user's hand posture. *UnderPressure* uses the gyroscope and vibration motors to infer the amount of pressure exerted on the device and can differentiate between three levels of pressure. I then go on to discuss *ContextType*, a keyboard that builds on top of GripSense to develop keymaps specific to different hand postures that lead to improved typing performance. Finally, I discuss *SwitchBack*, which improves reading speed by reminding the users about their progress while reading a body of text in the presence of

distractions. I also discuss various challenges that these systems face and must overcome before they can be seamlessly incorporated in daily usage.

I then highlight the sensors' output and demonstrate how they can improve the way a user interacts with multiple computing devices in the same environment. I discuss *SurfaceLink*, a system that uses the shared surface between devices as a bounded communication medium. It provides an end-to-end solution for impromptu, scalable interaction between devices using the on-device accelerometer, vibration motor, and speakers, as well as an off-device contact microphone. SurfaceLink detects devices on the same surface, enables a variety of continuous multi-touch gestures, and infers relative device arrangement. The evaluation also demonstrates that SurfaceLink scales well to increasing numbers of devices and users and can be an attractive interaction system as the number of devices or our interaction with these devices increases. I then briefly discuss *DopLink* and *AirLink*. These efforts use in-air gestures to improve multi-device interaction. This is especially important because there are many co-located devices that do not share the same surface. These efforts present a great opportunity to improve the way we interact with multiple devices in our environment. However, these sensors can be incredibly noisy; they are affected by many little changes in position, sound, *etc.* In many cases, multiple sensors need to inform each other and combine their inferences (much like the sensors in GripSense). SurfaceLink also combines the inferences from the accelerometer and microphone to listen to the user's knocks or motor-induced vibrations.

I then highlight the opportunities and challenges of indirect mobile sensing for health applications. I discuss these issues not only from an initial prototyping and evaluation perspective, but also through the lens of longitudinal real-world deployments. Here, I discuss *SpiroSmart*, a lung function assessment tool using the smartphone's built-in microphone. This system is tested in multiple deployments over the last five years. The first study was an in-depth feasibility investigation of using the microphone to assess a user's lung function, the challenges associated with it, and various design trade-offs. This study was instrumental in the development of the first complete SpiroSmart prototype that was then used in multiple clinics in the Seattle area for clinical validity. In these

clinical trials, the application was used by children and adults to investigate the effectiveness of SpiroSmart in a diverse set of physiologies and lung functions. These studies demonstrated a number of technical and usability challenges. These challenges were then overcome using an intelligent combination of on-device sensors and changes to the user interface. The updated system was then deployed at a larger scale in the Seattle area (USA), Pune (India), and Khulna (Bangladesh).

In an effort to make lung function assessment more accessible, I discuss *SpiroCall*, a call-in service that builds on top of SpiroSmart's algorithms. Here, the user does not need to own a smartphone. They can simply dial a 1-800 number from *any* phone and send their audio data over the phone call. The system then communicates the results back to them either through a text message or automated voice call. Eliminating the need of a smartphone is very powerful since it allows a much larger population to access the system; however, it comes with a number of technical and usability challenges. In order to alleviate some of these challenges, I describe a whistle design that can be used as a mouthpiece while performing spirometry. SpiroCall and the whistle were evaluated in a controlled evaluation with 50 patients and was found to have a very comparable performance to SpiroSmart. Similar to SpiroSmart, SpiroCall needs to be tested more deeply in the field to access the various challenges associated with it.

Driven by the challenges faced during the development, evaluation, and deployment of various explorations discussed in preceding chapters, particularly SpiroSmart, I then highlight a number of challenges associated with on-device sensors. I also discuss some recommendations for future mobile devices. The idea behind these recommendations is to make indirect mobile sensing more potent and uniform across usage scenarios and devices. I divide the improvements into three different categories:

1. To make the applications generalizable across devices.
2. To improve the performance of current applications.
3. Add new capabilities to the existing sensors.

The thesis includes a detailed discussion on how the sensors can evolve in future.

Currently, the sensors are optimized in a way that they mimic human capabilities: the microphone has a cutoff frequency around 19 kHz and the camera is sensitive to the visible spectrum. As we start thinking of these devices more as sensors, we start to go beyond the human capabilities. By exploiting ranges outside the human sensitivity, one can develop something that does not interfere with the user's senses and environment. Interference is not the only reason for going outside the range of human sensitivity. There are many phenomena around us that we are unable to sense. It would be great to have a tool that helps us observe phenomena occurring outside our sensitivity range, instead of just helping us document what we can already observe. I use the examples of *BiliCam* and *HyperCam* to motivate this idea. BiliCam uses the phone's camera to estimate bilirubin levels in newborns. Although the change in bilirubin level leads to a visible change to the baby's skin tone, it is difficult for humans to reach a precise and accurate estimate of bilirubin measurement from observation alone. HyperCam extends the capability of the camera to see *beyond* the human capabilities. The HyperCam device is a low-cost and adaptive hyperspectral camera. It serves as a demonstration that sensors are already very close to going beyond the human perception. If and when it actually happens, it will raise a number of usability challenges and the onus will be on application researchers and designers to make these technologies more seamless and useful.

Bibliography

[1] Chronic disease prevention and health promotion. http://www.cdc.gov/chronicdisease/overview/index.htm, April 2014.

[2] Abowd, G. D. What next, ubicomp?: celebrating an intellectual disappearing act. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (2012), 31–40.

[3] Adib, F., Mao, H., Kabelac, Z., Katabi, D., and Miller, R. C. Smart Homes that Monitor Breathing and Heart Rate. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15* (2015), 837–846.

[4] Agu, E., Pedersen, P., Strong, D., Tulu, B., He, Q., Wang, L., and Li, Y. The smartphone as a medical device: Assessing enablers, benefits and challenges. *2013 IEEE International Workshop of Internet-of-Things Networking and Control, IoT-NC 2013* (2013), 48–52.

[5] Al Faraj, K., Mojahid, M., and Vigouroux, N. BigKey: A Virtual Keyboard for Mobile Devices. In *Human-Computer Interaction. Ambient, Ubiquitous and Intelligent Interaction*, J. Jacko, Ed., vol. 5612 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2009, 3–10.

[6] Alani, H., O'Hara, K., and Shadbolt, N. Common features of killer apps: A comparison with Protégé.

[7] Allen, J., Gehrke, T., O'Sullivan, J. J., King, S. T., and Murray, A. Characterization of the Korotkoff sounds using joint time–frequency analysis. *Physiological Measurement 25*, 1 (feb 2004), 107–117.

[8] Alshaer, Hisham, Geoffrey R. Fernie, and Bradley., T. D. Phase tracking of the breathing cycle in sleeping subjects by frequency analysis of acoustic data. *International Journal of Healthcare Technology and Management 11.3* (2010), 163–175.

[9] Amft, O., Stäger, M., Lukowicz, P., and Tröster, G. *Analysis of chewing sounds for dietary monitoring*, vol. 3660 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, jan 2005.

[10] Anokwa, Y., Hartung, C., Brunette, W., Borriello, G., and Lerer, A. Open Source Data Collection in the Developing World, 2009.

[11] Arentz, W. A., and Bandara, U. Near ultrasonic directional data transfer for modern smartphones. In *Proc. UbiComp'11*, ACM Press (New York, New York, USA, 2011), 481.

[12] Aumi, M., Gupta, S., and Goel, M. DopLink: using the doppler effect for multi-device interaction. *Proceedings of the 2013 …* (2013), 583–586.

[13] Azenkot, S., and Zhai, S. Touch Behavior with Different Postures on Soft Smartphone Keyboards. In *Proc.. MobileHCI'2012* (2012).

[14] Bales, E., Nikzad, N., Quick, N., Ziftci, C., Patrick, K., and Griswold, W. Citisense: Mobile Air Quality Sensing for Individuals and Communities. Design and deployment of the Citisense mobile air-quality system. *Proceedings of the 6th International Conference on Pervasive Computing Technologies for Healthcare* (2012), 3–6.

[15] Balfanz, D., Smetters, D., Stewart, P., and Wong, H. Talking to strangers: Authentication in ad-hoc wireless networks. *Proceedings of the 9th Annual Network and Distributed System Security Symposium (NDSS) 2002* (2002), 7–19.

[16] Barabási, A.-L. Scale-free networks: a decade and beyond. *Science (New York, N.Y.) 325*, 5939 (2009), 412–413.

[17] Barnard, L., Yi, J. S., Jacko, J. a., and Sears, A. Capturing the effects of context on human performance in mobile computing systems. *Personal and Ubiquitous Computing 11*, 2 (2006), 81–96.

[18] Baronti, S., Casini, a., Lotti, F., and Porcinai, S. Multispectral Imaging System for the Mapping of Pigments in Works of Art by use of Principal-Component Analysis. *Applied optics 37*, 8 (mar 1998), 1299–309.

[19] Bedri, A., Verlekar, A., Thomaz, E., Avva, V., and Starner, T. A Wearable System for Detecting Eating Activities with Proximity Sensors in the Outer Ear. In *Int. Symp. on Wearable Computers (ISWC)* (2015), 91–92.

[20] Bishara, W., Su, T.-W., Coskun, A. F., and Ozcan, A. Lensfree on-chip microscopy over a wide field-of-view using pixel super-resolution. *Optics express 18*, 11 (may 2010), 11181–91.

[21] Boulanger, C., Boulanger, a., de Greef, L., Kearney, a., Sobel, K., Transue, R., Sweedyk, Z., Dietz, P., and Bathiche, S. Stroke rehabilitation with a sensing surface. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems-CHI '13* (2013), 1243.

[22] Bourouis, a., Feham, M., Hossain, M. a., and Zhang, L. An intelligent mobile based decision support system for retinal disease diagnosis. *Decision Support Systems 59*, 1 (2014), 341–350.

[23] Bourouis, A., Zerdazi, A., Feham, M., and Bouchachia, A. M-health: Skin disease analysis system using smartphone's camera. *Procedia Computer Science 19* (2013), 1116–1120.

[24] Bragdon, A., Nelson, E., Li, Y., and Hinckley, K. Experimental analysis of touch-screen gesture designs in mobile environments. *Proceedings of the 2011 annual conference on Human factors in computing systems CHI 11 17*, 2 (2011), 403–412.

[25] Brewster, S. Overcoming the lack of screen space on mobile computers. *Personal and Ubiquitous Computing 6*, 3 (2002), 188–205.

[26] Brouwer, A. F. J., Roorda, R. J., and Brand, P. L. P. Home spirometry and asthma severity in children. *European Respiratory Journal 28*, 6 (2006), 1131–1137.

[27] Brunette, W., Sodt, R., Chaudhri, R., Goel, M., Falcone, M., Van Orden, J., and Borriello, G. Open data kit sensors: a sensor integration framework for android at the application-level. *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12* (2012), 351.

[28] Brunette, W., Sundt, M., Dell, N., Chaudhri, R., Breit, N., and Borriello, G. Open Data Kit 2 . 0 : Expanding and Refining Information Services for Developing Regions. *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications - HotMobile '13* (2013), 6.

[29] Burke, J., Estrin, D., Hansen, M., Parker, a., Ramanathan, N., Reddy, S., and Srivastava, M. B. Participatory Sensing. 1–5.

[30] Butler, A., Izadi, S., and Hodges, S. SideSight: multi-"touch" interaction around small devices. *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology* (2008), 201–204.

[31] Campbell, A. T., Eisenman, S. B., Lane, N. D., Miluzzo, E., and Peterson, R. a. People-centric urban sensing. *Proceedings of the 2nd annual international workshop on Wireless internet - WICON '06* (2006), 18–31.

[32] Carspecken, C. W., Arteta, C., and Clifford, G. D. Telespiro: A low-cost mobile spirometer for resource-limited settings. In *Point-of-Care Healthcare Technologies (PHT), 2013 IEEE*, IEEE (2013), 144–147.

[33] Chang, Y. J., Chen, S. F., and Huang, J. D. A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. *Research in Developmental Disabilities 32*, 6 (2011), 2566–2570.

[34] Chaudhri, R., Brunette, W., Goel, M., Sodt, R., VanOrden, J., Falcone, M., and Borriello, G. Open data kit sensors. *Proceedings of the 2nd ACM Symposium on Computing for Development - ACM DEV '12* (2012), 1.

[35] Chaudhri, R., Vlachos, D., Kaza, J., Palludan, J., Bilbao, N., Martin, T., Borriello, G., Kolko, B., and Israel-Ballard, K. A system for safe flash-heat pasteurization of human breast milk. *Proceedings of the 5th ACM workshop on Networked systems for developing regions - NSDR '11* (2011), 9.

[36] Chen, K.-y., Ashbrook, D., Goel, M., Lee, S.-h., Patel, S., and Jose, S. AirLink: Sharing Files Between Multiple Devices Using In-Air Gestures. *UbiComp '14: Proceedings of the 16th International Conference on Ubiquitous Computing*, Figure 3 (2014).

[37] Chen, N.-C., Wang, K.-C., and Chu, H.-H. Listen-to-nose: a low-cost system to record nasal symptoms in daily life. 590–591.

[38] Chen, S. F., and Goodman, J. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, vol. 13 (1996), 310–318.

[39] Cheng, L.-P., Lee, M. H., Wu, C.-Y., Hsiao, F.-I., Liu, Y.-T., Liang, H.-S., Chiu, Y.-C., Lee, M.-S., and Chen, M. Y. IrotateGrasp: Automatic Screen Rotation Based on Grasp of Mobile Devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), 3051–3054.

[40] Choudhury, T., Hightower, J., Lamarca, A., Legrand, L., Rahimi, A., Rea, A., Hemingway, B., Koscher, K., Landay, J. a., Lester, J., and Wyatt, D. An embedded Activity Recognition system. *IEEE Pervasive Computing 7*, 2 (2008), 32–41.

[41] Chu, D., Lane, N. D., Lai, T. T.-T., Pang, C., Meng, X., Guo, Q., Li, F., and Zhao, F. Balancing energy, latency and accuracy for mobile sensor data classification. *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems - SenSys '11* (2011), 54.

[42] Clarkson, E. C., Patel, S. N., Pierce, J. S., and Abowd, G. D. Exploring Continuous Pressure Input for Mobile Phones. *UIST*, GIT-GVU-06-20 (2006), 1–4.

[43] Cochrane, G. M., Prieto, F., and Clark, T. J. Intrasubject variability of maximal expiratory flow volume curve. *Thorax 32*, 2 (1977), 171–6.

[44] Coffey, V. C. Multispectral Imaging Moves into the Mainstream. *OPN Optics & Photonics News*, 18–24.

[45] Consolvo, S., Everitt, K., Smith, I., and Landay, J. a. Design requirements for technologies that encourage physical activity. *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI '06* (2006), 457–466.

[46] Consolvo, S., McDonald, D. W., Toscos, T., Chen, M. Y., Froehlich, J., Harrison, B., Klasnja, P., LaMarca, A., LeGrand, L., Libby, R., Smith, I., and Landay, J. a. Activity sensing in the wild: a field trial of ubifit garden. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)* (2008), 1797–1806.

[47] Crapo, R. O., Hankinson, J. L., Irvin, C., and MacIntyre, N. R. Standardization of Spirometry. *American Journal of Respiratory and Critical Care Medicine*, 7 (1994).

[48] Crevoisier, A., and Polotti, P. Tangible Acoustic Interfaces and their Applications for the Design of New Musical Instruments. In *Proc. Nime'05* (2005), 97–100.

[49] Crossan, A., Murray-Smith, R., Brewster, S., Kelly, J., and Musizza, B. Gait Phase Effects in Mobile Interaction. *CHI '05 extended abstracts on Human factors in computing systems - CHI '05* (2005), 1312–1315.

[50] Cuypers, T., Francken, Y., Vanaken, C., Reeth, F. V., and Bekaert, P. Smartphone Localization on Interactive Surfaces Using the Built-in Camera. In *Proc. Procams'09*, vol. 2 (2009).

[51] Das, T., Mohan, P., Padmanabhan, V. N., Ramjee, R., and Sharma, A. PRISM: Platform for Remote Sensing using Smartphones. *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10* (2010), 63.

[52] Davis, N., and Sweeney, L. B. Apnea of infancy–a clinical problem. *The Western journal of medicine 144*, 4 (1986), 429–32.

[53] de Greef, L., Goel, M., Seo, M. J., Larson, E. C., Stout, J. W., Taylor, J. A., and Patel, S. N. BiliCam : Using Mobile Phones to Monitor Newborn Jaundice. In *Proceedings of UbiComp 2014.* (2014).

[54] Dearman, D., Guy, R. T., and Truong, K. N. Determining the Orientation of Proximate Mobile Devices using their Back Facing Camera. In *Proc. CHI'12*, no. Figure 2 (2012).

[55] Dell, N., and Borriello, G. Mobile Tools for Point-of-Care Diagnostics in the Developing World. In *DEV'12* (2012).

[56] Dell, N., Stevens, D., and Yager, P. Towards a Point-of-Care Diagnostic System : Automated Analysis of Immunoassay Test Data on a Cell Phone. *Networked Systems for Developing Regions*, June (2011), 3–8.

[57] Dietz, P., and Leigh, D. DiamondTouch: A Multi-User Touch Technology. In *Proceedings UIST 2001* (2001), 219–226.

[58] Eagle, N., and Pentland, A. Reality mining: Sensing complex social systems. *Personal and Ubiquitous Computing 10*, 4 (2006), 255–268.

[59] Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., Ishwaran, H., Knight, K., Loubes, J. M., Massart, P., Madigan, D., Ridgeway, G., Rosset, S., Zhu, J. I., Stine, R. A., Turlach, B. A., Weisberg, S., Hastie, T., Johnstone, I., and Tibshirani, R. Least angle regression. *Annals of Statistics 32*, 2 (2004), 407–499.

[60] Eisner, M. D., Anthonisen, N., Coultas, D., Kuenzli, N., Perez-Padilla, R., Postma, D., Romieu, I., Silverman, E. K., and Balmes, J. R. An official american thoracic society public policy statement: Novel risk factors and the global burden of chronic obstructive pulmonary disease. *American journal of respiratory and critical care medicine 182*, 5 (2010), 693–718.

[61] Epstein, D. A., Borning, A., and Fogarty, J. Fine-grained sharing of sensed physical activity: A value sensitive approach. In *Proceedings of the 2013 ACM International*

Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '13, ACM (New York, NY, USA, 2013), 489–498.

[62] Ericsson. Ericsson Mobility Report: On the Pulse of the Networked Society. Tech. rep., 2015.

[63] Essl, G., Rohs, M., and Kratz, S. Use the Force (or something) - Pressure and Pressure - Like Input for Mobile Music Performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression* (2010), 182–185.

[64] Findlater, L., and Wobbrock, J. Personalized Input: Improving Ten-Finger Touchscreen Typing through Automatic Adaptation. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12* (2012), 815–824.

[65] Findlater, L., Wobbrock, J. O., and Wigdor, D. Typing on flat glass: examining ten-finger expert typing patterns on touch surfaces. *CHI '11 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2011), 2453–2462.

[66] Finkelstein, J., Cabrera, M. R., and Hripcsak, G. Internet-based home asthma telemonitoring: Can patients handle the technology? *Chest 117*, 1 (2000), 148–155.

[67] Fitzmaurice, G., Ishii, H., and Buxton, W. Bricks: laying the foundations for graspable user interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems* (1995), 442–449.

[68] Fitzpatrick, K., Brewer, M., and Turner, S. Another Look at Pedestrian Walking Speed. *Transportation Research Record 1982*, 1 (2006), 21–29.

[69] Flanagan, J. L. *Speech analysis synthesis and perception*, vol. 3. Springer Science & Business Media, 2013.

[70] Franko, O., Bray, C., and Newton, P. Validation of a scoliometer smartphone app to assess scoliosis. *Journal of pediatric orthopedics* (2012).

[71] Frey, H., and Sturm, P. Mark Weiser. *Syssoft.Uni-Trier.De* (1999).

[72] Georgiev, P., Lane, N. D., Rachuri, K. K., and Mascolo, C. DSP.Ear: Leveraging Co-Processor Support for Continuous Audio Sensing on Smartphones. *SenSys* (2014), 15.

[73] Go, K., and Endo, Y. CATKey : Customizable and Adaptable Touchscreen Keyboard with Bubble Cursor-Like Visual Feedback. *Ifip International Federation For Information Processing* (2007), 493–496.

[74] Goel, M., Whitmire, E., Mariakakis, A., Saponas, T. S., Joshi, N., Morris, D., Guenter, B., Gavriliu, M., Borriello, G., and Patel, S. N. Hypercam: hyperspectral imaging for ubiquitous computing applications. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM (2015), 145–156.

[75] Goltz, D. M., Cloutis, E., Norman, L., and Attas, M. Enhancement of Faint Text Using Visible (420–720 nm) Multispectral Imaging. *Restaurator 28*, 1 (jan 2007), 11–28.

[76] Goodman, J., Venolia, G., Steury, K., and Parker, C. Language modeling for soft keyboards. In *Proceedings of the 7th international conference on Intelligent user interfaces - IUI '02* (2002), 194–195.

[77] Grimaldi, D., Kurylyak, Y., Lamonaca, F., and Nastro, A. Photoplethysmography detection by smartphone's videocamera. *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS'2011 1*, September (2011), 488–491.

[78] Gunawardana, A., Paek, T., and Meek, C. Usability Guided Key-Target Resizing for Soft Keyboards. In *Proceedings of the 15th international conference on Intelligent user interfaces - IUI '10* (2010), 111–118.

[79] Gunawardana, A., Paek, T., and Meek, C. Usability guided key-target resizing for soft keyboards. In *Proceedings of the 15th international conference on Intelligent user interfaces*, IUI '10, ACM (New York, NY, USA, 2010), 111–118.

[80] Gupta, M. R., Garcia, E. K., and Chin, E. Adaptive local linear regression with application to printer color management. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society 17*, 6 (2008), 936–945.

[81] Gupta, S., Chang, P., Anyigbo, N., and Sabharwal, A. mobileSpiro. In *Proceedings of the First ACM Workshop on Mobile Systems, Applications, and Services for Healthcare - mHealthSys '11*, ACM Press (New York, New York, USA, nov 2011), 1.

[82] Gupta, S., Morris, D., Patel, S., and Tan, D. SoundWave: Using the Doppler Effect to Sense Gestures. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12* (2012), 1911–1914.

[83] Hankinson, J. L., Odencrantz, J. R., and Fedan, K. B. Spirometric reference values from a sample of the general us population. *American journal of respiratory and critical care medicine 159*, 1 (1999), 179–187.

[84] Harrison, B. L., Fishkin, K. P., Gujar, A., Mochon, C., and Want, R. Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. In *Proc. CHI 1998.*, CHI '98, ACM Press (New York, NY, USA, 1998), 17–24.

[85] Harrison, C., and Hudson, S. E. Scratch Input : Creating Large, Inexpensive, Unpowered and Mobile Finger Input Surfaces. In *Proc. UIST 2008* (2008).

[86] Harrison, C., Schwarz, J., and Hudson, S. E. TapSense : Enhancing Finger Interaction on Touch Surfaces. In *Proc. UIST 2011* (2011).

[87] Harrison, C., Xiao, R., and Hudson, S. E. Acoustic Barcodes : Passive , Durable and Inexpensive Notched Identification Tags. In *Proc. UIST 2012* (2012), 1–5.

[88] Hart, S. G. Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting 50*, 9 (oct 2006), 904–908.

[89] Heinemann, L., and Schmelzeisen-Redeker, G. Non-invasive continuous glucose monitoring in type i diabetic patients with optical glucose sensors. *Diabetologia 41*, 7 (1998), 848–854.

[90] Heo, S., and Lee, G. Force Gestures : Augmented Touch Screen Gestures Using Normal and Tangential Force. *Proceedings of the 2011 annual conference on Human factors in computing systems* (2011), 1909–1914.

[91] Heo, S., and Lee, G. ForceTap: Extending the Input Vocabulary of Mobile Touch Screens by adding Tap Gestures. *MobileHCI* (2011), 113–122.

[92] Himberg, J., Häkkilä, J., Kangas, P., and Mäntyjärvi, J. On-line personalization of a touch screen based keyboard. *Proceedings of the 8th international conference on Intelligent user interfaces - IUI '03* (2003), 77.

[93] Hinckley, K., and Horvitz, E. Toward more sensitive mobile phones. *Proceedings of the 14th annual ACM symposium on User interface software and technology - UIST '01 3*, 2 (2001), 191.

[94] Hinckley, K., Pierce, J., Sinclair, M., and Horvitz, E. Sensing techniques for mobile interaction. In *Proc. UIST 2000*, UIST '00, ACM (New York, NY, USA, 2000), 91–100.

[95] Hinckley, K., and Ramos, G. Cooperative Stitching : Spontaneous Wireless Connections for Small Co-Located Groups (TECHNOTE). 1–4.

[96] Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P., and Smith, M. Stitching : Pen Gestures that Span Multiple Displays. In *Proc. AVI'04* (2004), 23–31.

[97] Hinckley, K., and Song, H. Sensor synaesthesia: touch in motion, and motion in touch. In *Proc. CHI 2011*, CHI '11, ACM (New York, NY, USA, 2011), 801–810.

[98] Holmquist, L. E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., and Gellersen, H.-w. Smart-Its Friends : A Technique for Users to Easily Establish Connections between Smart Artefacts. In *Proc. UbiComp' 01* (2001).

[99] Holz, C., and Baudisch, P. Understanding touch. In *Proc. CHI 2011*, CHI '11, ACM (New York, NY, USA, 2011), 2501–2510.

[100] Homs-Corbera, A., Fiz, J. A., Morera, J., and Jané, R. Time-frequency detection and analysis of wheezes during forced exhalation. *IEEE transactions on bio-medical engineering 51*, 1 (jan 2004), 182–6.

[101] Hutama, W., Song, P., Fu, C., and Goh, W. Distinguishing multiple smart-phone interactions on a multi-touch wall display using tilt correlation. In *CHI 2011* (2011), 3315–3318.

[102] Hutama, W., Song, P., Fu, C.-W., and Goh, W. B. Distinguishing multiple smart-phone interactions on a multi-touch wall display using tilt correlation. In *Proc. CHI'11*, ACM Press (New York, New York, USA, 2011), 3315–3318.

[103] Iwasaki, K., Miyaki, T., and Rekimoto, J. Expressive Typing: A New Way to Sense Typing Pressure and Its Applications. *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems - CHI EA '09* (2009), 4369–4374.

[104] Johnson, P. Usability and mobility; interactions on the move. In *First Workshop on Human-Computer Interaction with Mobile Devices.* (1998).

[105] Kaiser, S., Parks, A., Leopard, P., Albright, C., Carlson, J., Goel, M., Nassehi, D., and Larson, E. C. Design and Learnability of Vortex Whistles for Managing Chronic Lung Function via Smartphones. In *Proc. UbiComp'16* (2016).

[106] Kane, S. K., Avrahami, D., Wobbrock, J. O., Harrison, B., Rea, A. D., Philipose, M., and Lamarca, A. Bonfire : A Nomadic System for Hybrid Laptop-Tabletop Interaction. In *Proc. UIST 2009* (2009), 129–138.

[107] Kane, S. K., Wobbrock, J. O., Harniss, M., and Johnson, K. L. TrueKeys: identifying and correcting typing errors for people with motor impairments. *Proceedings IUI'08*, 1 (2008), 349–352.

[108] Kane, S. K., Wobbrock, J. O., and Smith, I. E. Getting off the treadmill: evaluating walking user interfaces for mobile devices in public spaces. *Interfaces* (2008), 109–118.

[109] Kanjo, E., Bacon, J., Landshoff, P., and Roberts, D. MobSens: Making Smart Phones Smarter. *IEEE Pervasive Computing* (2009).

[110] Kansal, A., Saponas, S., Brush, a. B., McKinley, K. S., Mytkowicz, T., and Ziola, R. The latency, accuracy, and battery (LAB) abstraction: programmer productivity and energy efficiency for continuous mobile context sensing. *Proc. of OOPSLA'13* (2013), 661–676.

[111] Kargel, J. S., Abrams, M. J., Bishop, M. P., Bush, A., Hamilton, G., Jiskoot, H., Kääb, A., Kieffer, H. H., Lee, E. M., Paul, F., Rau, F., Raup, B., Shroder, J. F., Soltesz, D., Stainforth, D., Stearns, L., and Wessels, R. Multispectral imaging contributions to global land ice measurements from space. *Remote Sensing of Environment 99*, 1-2 (nov 2005), 187–219.

[112] Karlson, A. K., and Bederson, B. B. Understanding Single-Handed Mobile Device Interaction. Tech. rep., 2006.

[113] Karlson, A. K., Bederson, B. B., and SanGiovanni, J. AppLens and launchTile: two designs for one-handed thumb use on small devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, ACM (New York, NY, USA, 2005), 201–210.

[114] Kaufmann, B., and Buechley, L. Amarino. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services - MobileHCI '10* (2010), 291.

[115] Kessler, R., Stáhl, E., Vogelmeier, C., Haughney, J., Trudeau, E., Löfdahl, C. G., and Partridge, M. R. Patient understanding, detection experience of COPD exacerbations: An observational, interview-based study. *Chest 130*, 1 (2006), 133–142.

[116] Kim, K.-E., Chang, W., Cho, S.-J., Shim, J., Lee, H., Park, J., Lee, Y., and KimSangryong. Hand Grip Pattern Recognition for Mobile User Interfaces. In *Proc. AAAI'06* (2006).

[117] Kim, Y.-J., and Yoon, G. Prediction of glucose in whole blood by near-infrared spectroscopy: influence of wavelength region, preprocessing, and hemoglobin concentration. *Journal of biomedical optics 11*, 4 (2013), 041128.

[118] Klasnja, P., and Pratt, W. Healthcare in the pocket: Mapping the space of mobile-phone health interventions. *Journal of Biomedical Informatics 45*, 1 (2012), 184–198.

[119] Knudson, R. J., Slatin, R. C., Lebowitz, M. D., and Burrows, B. The maximal expiratory flow-volume curve. Normal standards, variability, and effects of age. *The American review of respiratory disease 113*, 5 (may 1976), 587–600.

[120] Kortuem, G., Kray, C., and Gellersen, H. Sensing and visualizing spatial relations of mobile devices. In *Proc. UIST'05*, ACM Press (New York, New York, USA, 2005), 93.

[121] Kristensson, P.-O., and Zhai, S. SHARK^2: A Large Vocabulary Shorthand Writing System for Pen-Based Computers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology - UIST '04* (2004), 43–52.

[122] Kristensson, P.-O., and Zhai, S. Relaxing Stylus Typing Precision by Geometric Pattern Matching. In *Proceedings of the 10th international conference on Intelligent user interfaces - IUI '05* (2005), 151–158.

[123] Kroutil, J., Laposa, A., and Husak, M. Respiration monitoring during sleeping. In *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies - ISABEL '11*, ACM Press (New York, New York, USA, oct 2011), 1–5.

[124] Kucera, H., and Francis, W. N. Computational Analysis of Present-Day American English. *American Documentation 19*, 4 (1968), 419.

[125] Kung, H.-C., Hoyert, D. L., Xu, J., Murphy, S. L., et al. Deaths: final data for 2005. *National Vital Statistics Report 56*, 10 (2008), 1–120.

[126] Kunze, K., and Lukowicz, P. Symbolic object localization through active sampling of acceleration and sound signatures. *9th International Conference on Ubiquitous Computing - UbiComp '07* (2007), 163–180.

[127] Kunzli, N., Ackermann-Liebrich, U., Keller, R., Perruchoud, A. P., Schindler, C., Leuenberger, P., Alean, P., Blaser, K., Bolognini, G., Bongard, J. P., Braendli, O., Braun, P., Bron, C., Brutsche, M., Defila, C., Domenighetti, G., Elsasser, S., Filliger,

P., and Grize, L. Variability of FVC and FEV1 due to technician, team, device and subject in an eight centre study: Three quality control studies in SAPALDIA. In *European Respiratory Journal*, vol. 8 (1995), 371–376.

[128] Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, vol. 1 (2001), 282–289.

[129] Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., and Campbell, A. T. A survey of mobile phone sensing. *IEEE Communications Magazine 48*, 9 (2010), 140–150.

[130] Larson, E. C., Goel, M., Boriello, G., Heltshe, S., Rosenfeld, M., and Patel, S. N. SpiroSmart : Using a Microphone to Measure Lung Function on a Mobile Phone. In *UbiComp'12*, no. Figure 1 (2012).

[131] Larson, E. C., Lee, T., Liu, S., Rosenfeld, M., and Patel, S. N. Accurate and Privacy Preserving Cough Sensing using a Low-Cost Microphone. In *Proc. UbiComp'11* (2011).

[132] Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabási, A.-L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D., and van Alstyne, M. Computational social science. *Science 323*, February (2009), 721–723.

[133] Lee-Chiong, T. L., and Magalang, U. Monitoring respiration during sleep. *Respiratory Care Clinics of North America 11*, 4 (2005), 663–678.

[134] Lester, J., Choudhury, T., and Borriello, G. A Practical Approach to Recognizing Physical Activities. *Pervasive Computing 3968* (2006), 1–16.

[135] Levenson, R. M., and Mansfield, J. R. Multispectral Imaging in Biology and Medicine : Slices of Life. *2006 International Society for Analytical Cytology 758* (2006), 748–758.

[136] LiKamWa, R., Hou, Y., Gao, J., Polansky, M., and Zhong, L. Redeye: Analog convnet image sensor architecture for continuous mobile vision.

[137] Lin, F. X., Rahmati, A., and Zhong, L. Dandelion: A Framework for Transparently Programming Phone-Centered Wireless Body Sensor Applications for Health. *Wireless Health 2010* (2010), 74–83.

[138] Lin, F. X., Wang, Z., LiKamWa, R., and Zhong, L. Reflex: Using Low-Power Processors in Smartphones without Knowing Them. *Proceedings of ASPLOS'12 40*, 1 (2012), 13.

[139] Lin, M., Goldman, R., Price, K. J., Sears, A., and Jacko, J. How do people tap when walking? An empirical investigation of nomadic data entry. *International Journal of Human Computer Studies 65*, 9 (2007), 759–769.

[140] Lopes, P., Jota, R., and Jorge, J. A. Augmenting touch interaction through acoustic sensing. In *Proc. ITS '11*, ACM Press (New York, New York, USA, 2011), 53.

[141] Lu, H., Bernheim Brush, A. J., Priyantha, B., Karlson, A. K., and Liu, J. SpeakerSense: Energy efficient unobtrusive speaker identification on mobile phones. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6696 LNCS (2011), 188–205.

[142] Lu, H., Frauendorfer, D., Rabbi, M., Mast, M. S., Chittaranjan, G. T., Campbell, A. T., Gatica-Perez, D., and Choudhury, T. StressSense. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12* (2012), 351.

[143] Lu, H., Pan, W., Lane, N., Choudhury, T., and Campbell, A. SoundSense: scalable sound sensing for people-centric applications on mobile phones. *Proceedings of the 7th international conference on Mobile systems, applications, and services* (2009), 165–178.

[144] Lu, R. Predicting Firmness and Sugar Content of Sweet Cherries Using Near-Infrared Diffuse Reflectance Spectroscopy. *Transactions of the American Society of Agricultural Engineers 44*, 5 (2001), 1265–1271.

[145] Lu, R. Multispectral imaging for predicting firmness and soluble solids content of apple fruit. *Postharvest Biology and Technology 31*, 2 (feb 2004), 147–157.

[146] Lucero, A., Jokela, T., Palin, A., Aaltonen, V., and Nikara, J. EasyGroups : Binding Mobile Devices for Collaborative Interactions. In *Proc. CHI'12 (WIP)* (2012), 2189–2194.

[147] MacKay, B., Dearman, D., Inkpen, K., and Watters, C. Walk 'n scroll: a comparison of software-based navigation techniques for different levels of mobility. *Proceedings of the 7th International Conference on Human-Computer Interaction with Mobile Devices and Services* (2005), 183–190.

[148] MacKenzie, I. S., and Soukoreff, R. W. Text Entry for Mobile Computing: Models and Methods,Theory and Practice. *Human–Computer Interaction 17*, 2-3 (2002), 147–198.

[149] MacKenzie, I. S., and Soukoreff, R. W. Phrase sets for evaluating text entry techniques. In *CHI '03 extended abstracts on Human factors in computing systems*, CHI EA '03, ACM (New York, NY, USA, 2003), 754–755.

[150] Majchrzak, T. a., and Chakravorty, A. Improving the compliance of transplantation medicine patients with an integrated mobile system. *Proceedings of the Annual Hawaii International Conference on System Sciences* (2011), 2696–2705.

[151] Mariakakis, A., Goel, M., Aumi, M. T. I., Patel, S. N., and Wobbrock, J. O. SwitchBack: Using Focus and Saccade Tracking to Guide Users' Attention for Mobile Task Resumption. *Proceedings of the ACM CHI'15 Conference on Human Factors in Computing Systems 1* (2015), 2953–2962.

[152] Matsushita, N., and Rekimoto, J. HoloWall: Designing a Finger; Hand, Body, and Object Sensitive Wall. In *Proc. UIST'97* (1997), 209–210.

[153] Mattila, E., Pärkkä, J., Hermersdorf, M., Kaasinen, J., Vainio, J., Samposalo, K., Merilahti, J., Kolari, J., Kulju, M., Lappalainen, R., and Korhonen, I. Mobile diary for wellness management - Results on usage and usability in two user studies. *IEEE Transactions on Information Technology in Biomedicine 12*, 4 (2008), 501–512.

[154] Mayrhofer, R., and Gellersen, H. Shake Well Before Use : Authentication Based on Accelerometer Data. In *Proc. Pervasive'07* (2007), 144–161.

[155] Miller, M. R., Hankinson, J., Brusasco, V., Burgos, F., Casaburi, R., Coates, A., Crapo, R., Enright, P., van der Grinten, C. P. M., Gustafsson, P., Jensen, R., Johnson, D. C., MacIntyre, N., McKay, R., Navajas, D., Pedersen, O. F., Pellegrino, R., Viegi, G., and Wanger, J. Standardisation of spirometry. *The European respiratory journal 26*, 2 (aug 2005), 319–38.

[156] Miluzzo, E. Smartphone Sensing. *ProQuest Dissertations and Theses* (2011), 148.

[157] Mizobuchi, S., Chignell, M., and Newton, D. 140427_MobileHCI '05_Mobile text entry- relationship between walking speed and text input task difficulty. *the 7th international conference on Human computer interaction with mobile devices & services* (2005), 122–128.

[158] Monsell, S. Task switching, 2003.

[159] Morgan, D. R., and Zierdt, M. G. Novel signal processing techniques for Doppler radar cardiopulmonary sensing. *Signal Processing 89*, 1 (2009), 45–66.

[160] Munson, S., and Consolvo, S. Exploring Goal-setting, Rewards, Self-monitoring, and Sharing to Motivate Physical Activity. *Proceedings of the 6th International Conference on Pervasive Computing Technologies for Healthcare* (2012), 25–32.

[161] Murray, C. J., and Lopez, A. D. Alternative projections of mortality and disability by cause 1990-2020: Global burden of disease study. *The Lancet 349*, 9064 (1997), 1498–1504.

[162] Murray-Smith, R., Williamson, J., Hughes, S., and Quaade, T. Stane : Synthesized Surfaces for Tactile Input. In *Proc. CHI'08* (2008).

[163] Mustonen, T., Olkkonen, M., and Hakkinen, J. Examining Mobile Phone Text Legibility while Walking. *Extended abstracts of the 2004 conference on Human factors and computing systems - CHI '04* (2004), 1243–1246.

[164] Neuman, M. R. Vital signs: heart rate. *IEEE pulse 1*, 3, 51–5.

[165] Nikzad, N., Verma, N., Ziftci, C., Bales, E., Nichole, Q., Piero, Z., Patrick, K., Dasgupta, S., Krueger, I., Rosing, T. S., and Griswold, W. G. CitiSense: Improving Geospatial Environmental Assessment of Air Quality Using a Wireless Personal Exposure Monitoring System. In *Proceedings of the Conference on Wireless Health* (2012), 1–8.

[166] Nirjon, S., Dickerson, R. F., Asare, P., Li, Q., Hong, D., Stankovic, J. a., Hu, P., Shen, G., and Jiang, X. Auditeur: A Mobile-cloud Service Platform for Acoustic Event Detection on Smartphones. *Proc. of MobiSys* (2013), 403.

[167] Nishimura, J. Eating habits monitoring using wireless wearable in-ear microphone. In *2008 3rd International Symposium on Wireless Pervasive Computing*, IEEE (may 2008), 130–132.

[168] Ölmez, T., and Dokur, Z. Classification of heart sounds using an artificial neural network. *Pattern Recognition Letters 24* (2003), 617–629.

[169] Otulana, B. A., Higenbottam, T., Ferrari, L., Scott, J., Igboaka, G., and Wallwork, J. The use of home spirometry in detecting acute lung rejection and infection following heart-lung transplantation. *Chest 97*, 2 (1990), 353–357.

[170] Pamplona, V. F., Mohan, A., Oliveira, M. M., and Raskar, R. Netra. *ACM Transactions on Graphics 29*, 4 (2010), 1.

[171] Paradiso, J. a., and Leo, C. K. Tracking and characterizing knocks atop large interactive displays. *Sensor Review 25*, 2 (2005), 134–143.

[172] Paradiso, J. A., Leo, C. K., Checka, N., and Hsiao, K. Passive acoustic sensing for tracking knocks atop large interactive displays. *Proc. IEEE Sensors 2002 1* (2002), 521–527.

[173] Pasterkamp, H., Kraman, S. S., and Wodicka, G. R. Respiratory sounds: Advances beyond the stethoscope, 1997.

[174] Pesola, G. R., O'Donnell, P., Pesola, G. R., Chinchilli, V. M., and Saari, A. F. Peak expiratory flow in normals: comparison of the mini Wright versus spirometric predicted peak flows. *The Journal of asthma : official journal of the Association for the Care of Asthma 46*, 8 (2009), 845–848.

[175] Poh, M. Z., Kim, K., Goessling, A. D., Swenson, N. C., and Picard, R. W. Heartphones: Sensor earphones and mobile application for non-obtrusive health monitoring. *Proceedings - International Symposium on Wearable Computers, ISWC* (2009), 153–154.

[176] Poh, M.-Z., McDuff, D. J., and Picard, R. W. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Optics express 18*, 10 (may 2010), 10762–74.

[177] Priyantha, B., Lymberopoulos, D., and Liu, J. Enabling energy efficient continuous sensing on mobile phones with LittleRock. *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks - IPSN '10* (2010), 420.

[178] Rahman, T., Adams, A., Zhang, M., Cherry, E., Zhou, B., Peng, H., and Choudhury, T. BodyBeat: a mobile system for sensing non-speech body sounds. In *MobiSys '14 Proceedings of the 12th annual international conference on Mobile systems, applications, and services* (2014), 2–13.

[179] Rahman, T., Adams, A. T., Zhang, M., Cherry, E., Zhou, B., Peng, H., and Choudhury, T. BodyBeat : A Mobile System for Sensing Non-Speech Body Sounds. 2–13.

[180] Rahman, Tauhidur, Adams, A. T., Ravichandran, R. V., Zhang, M., Patel, S. N., Kientz, J. A., and Choudhury, T. DoppleSleep: A contactless unobtrusive sleep sensing system using short-range Doppler radar. *UbiComp* (2015), 39–50.

[181] Rajalakshmi, R., Nandakumar, D., and Nathaniel, M. Contactless Sleep Apnea Detection on Smartphones. *The 13th Annual International Conference on Mobile Systems, Applications and Services* (2015), 45–57.

[182] Ramos, G., Boulos, M., and Balakrishnan, R. Pressure Widgets. *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04 6*, 1 (2004), 487–494.

[183] Ramos, G., Hinckley, K., Wilson, A., and Sarin, R. Synchronous Gestures in Multi-Display Environments. *Human-Computer Interaction, Special Issue: Ubiquitous Multi-Display Environments 24*, 1-2 (2009), 117–169.

[184] Rana, R. K., Chou, C. T., Kanhere, S. S., Bulusu, N., and Hu, W. Ear-Phone : An End-to-End Participatory Urban Noise Mapping System. In *Proceedings of the International Conference on Information Processing in Sensor Networks IPSN* (2010), 105–116.

[185] Ravichandran, R., Saba, E., Chen, K. Y., Goel, M., Gupta, S., and Patel, S. N. WiBreathe: Estimating respiration rate using wireless signals in natural settings in the home. In *2015 IEEE International Conference on Pervasive Computing and Communications, PerCom 2015* (2015), 131–139.

[186] Rebuck, D. A., Hanania, N. A., D'Urzo, A. D., and Chapman, K. R. The accuracy of a handheld portable spirometer. *Chest 109*, 1 (1996), 152–157.

[187] Reddy, S., Parker, A., Hyman, J., Burke, J., Estrin, D., and Hansen, M. Image Browsing , Processing , and Clustering for Participatory Sensing : Lessons From a DietSense Prototype. *EmNets'07* (2007), 13–17.

[188] Regan, E. A., Lynch, D. A., Curran-Everett, D., Curtis, J. L., Austin, J. H., Grenier, P. A., Kauczor, H.-U., Bailey, W. C., DeMeo, D. L., Casaburi, R. H., et al. Clinical and radiologic disease in smokers with normal spirometry. *JAMA internal medicine 175*, 9 (2015), 1539–1549.

[189] Rekimoto, J., Ayatsuka, Y., and Kohno, M. SyncTap: An interaction technique for mobile networking. *Human-Computer Interaction with Mobile Devices and Services* (2003), 104–115.

[190] Rekimoto, J., and Saitoh, M. Augmented surfaces: a spatially continuous work space for hybrid computing environments. *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit* (1999), 378–385.

[191] Rubini, A., Parmagnani, A., and Bondì, M. Daily variations in lung volume measurements in young healthy adults. *Biological Rhythm Research 42*, 3 (jun 2011), 261–265.

[192] Rubini, A., Parmagnani, A., Redaelli, M., Bondì, M., Del Monte, D., and Catena, V. Daily variations of spirometric indexes and maximum expiratory pressure in young healthy adults. *Biological Rhythm Research 41*, 2 (2010), 105–112.

[193] Rudchenko, D., Paek, T., and Badger, E. Text text revolution: A game that improves text entry on mobile touchscreen keyboards. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6696 LNCS (2011), 206–213.

[194] Sakka, E. J., Aggelidis, P., and Psimarnou, M. Mobispiro: A novel spirometer. In *XII Mediterranean Conference on Medical and Biological Engineering and Computing 2010*, Springer (2010), 498–501.

[195] Sashima, A., Inoue, Y., Ikeda, T., Yamashita, T., and Kurumatani, K. CONSORTS-S: A mobile sensing platform for context-aware services. In *ISSNIP 2008 - Proceedings of the 2008 International Conference on Intelligent Sensors, Sensor Networks and Information Processing* (2008), 417–422.

[196] Sato, H., and Watanabe, K. Experimental study on the use of a vortex whistle as a flowmeter. *IEEE Transactions on Instrumentation and Measurement 49*, 1 (2000), 200–205.

[197] Schildbach, B., and Rukzio, E. Investigating selection and reading performance on a mobile phone while walking. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services - MobileHCI '10* (2010), 93.

[198] Schmidt, A., Aidoo, K., Takaluoma, A., Tuomela, U., Van Laerhoven, K., and de Velde, W. Advanced Interaction in Context. In *Handheld and Ubiquitous Computing*, H.-W. Gellersen, Ed., vol. 1707 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1999, 89–101.

[199] Schmidt, D., Chehimi, F., Rukzio, E., and Gellersen, H. PhoneTouch : A Technique for Direct Phone Interaction on Surfaces. In *Proc. UIST'10* (2010).

[200] Sears, A., Lin, M., Jacko, J., and Xiao, Y. When Computers Fade Pervasive Computing and Situationally-Induced Impairments and Disabilities. *HCI International 2* (2003), 1298–1302.

[201] Seemungal, T. a., Donaldson, G. C., Bhowmik, a., Jeffries, D. J., and Wedzicha, J. a. Time course and recovery of exacerbations in patients with chronic obstructive pulmonary disease. *American journal of respiratory and critical care medicine 161*, 5 (2000), 1608–1613.

[202] Seniuk, A., and Blostein, D. Pen Acoustic Emissions for Text and Gesture Recognition. In *2009 10th International Conference on Document Analysis and Recognition*, Ieee (2009), 872–876.

[203] Sevick, M. A., Trauth, J. M., Ling, B. S., Anderson, R. T., Piatt, G. A., Kilbourne, A. M., and Goodman, R. M. Patients with complex chronic diseases: Perspectives on supporting self-management. In *Journal of General Internal Medicine*, vol. 22 (2007), 438–444.

[204] Shen, C., Chakraborty, S., Raghavan, K. R., Choi, H., and Srivastava, M. B. Exploiting processor heterogeneity for energy efficient context inference on mobile phones. *Proceedings of the Workshop on Power-Aware Computing and Systems - HotPower '13* (2013), 1–5.

[205] Shen, H., Balasubramanian, A., LaMarca, A., and Wetherall, D. Enhancing mobile apps to use sensor hubs without programmer effort. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM (2015), 227–238.

[206] Shen, L., Hagen, J. a., and Papautsky, I. Point-of-care colorimetric detection with a smartphone. *Lab on a Chip 12*, 21 (2012), 4240.

[207] Smith, M. O., Ustin, S. L., Adams, J. B., and Gillespie, A. R. Vegetation in deserts: I. A regional measure of abundance from multispectral images. *Remote Sensing of Environment 31*, 1 (1990), 1–26.

[208] Song, K.-Y., Oh, H.-J., Choi, J., Park, I., Lee, C., and Lee, S. Prediction of landslides using ASTER imagery and data mining models. *Advances in Space Research 49*, 5 (mar 2012), 978–993.

[209] Sorber, J., Banerjee, N., Corner, M., and Rollins, S. Turducken: hierarchical power management for mobile devices. *Proceedings of the 3rd international conference on Mobile systems, applications, and services* (2005), 261–274.

[210] Soukoreff, R. W., and MacKenzie, I. S. Metrics for text entry research- an evaluation of MSD and KSPC, and a new unified error metric. *Proceedings of the conference on Human factors in computing systems - CHI '03 5* (2003), 113–120.

[211] Sun, X., Lu, Z., Hu, W., and Cao, G. Symdetector: Detecting sound-related respiratory symptoms using smartphones. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, ACM (New York, NY, USA, 2015), 97–108.

[212] Swan, M. The Quantified Self. *Big Data 1*, 2 (2013).

[213] Swanney, M. P., Ruppel, G., Enright, P. L., Pedersen, O. F., Crapo, R. O., Miller, M. R., Jensen, R. L., Falaschetti, E., Schouten, J. P., Hankinson, J. L., et al. Using the lower limit of normal for the fev1/fvc ratio reduces the misclassification of airway obstruction. *Thorax* (2008).

[214] Taylor, B. T., and Bove Jr., V. M. Graspables: grasp-recognition as a user interface. In *Proc. CHI 2009*, CHI '09, ACM (New York, NY, USA, 2009), 917–926.

[215] Thiel, B., Kloch, K., and Lukowicz, P. Sound-based proximity detection with mobile phones. *Proc. PhoneSense'12* (2012), 1–4.

[216] Thomaz, E., Essa, I., and Abowd, G. A practical approach for recognizing eating moments with wrist-mounted inertial sensing. In *of the 2015 ACM International Joint …* (2015).

[217] Thomaz, E., Zhang, C., Essa, I., and Abowd, G. D. Inferring Meal Eating Activities in Real World Settings from Ambient Sounds: A Feasibility Study. *Iui* (2015), 427–431.

[218] Toscos, T., Faber, A., Connelly, K., and Upoma, A. M. Encouraging physical activity in teens Can technology help reduce barriers to physical activity in adolescent girls? *Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on 3*, Group 3 (2008), 218–221.

[219] Townsend, M. C. ACOEM position statement. Spirometry in the occupational setting. American College of Occupational and Environmental Medicine. *J Occup Environ Med 42*, 3 (2000), 228–245.

[220] Van Beers, R., Aernouts, B., De Baerdemaeker, J., and Saeys, W. Apple ripeness detection using hyperspectral laser scatter imaging. *Sensing Technologies for Biomaterial, Food, and Agriculture 8881* (may 2013).

[221] Van Laerhoven, K., and Cakmakci, O. What shall we teach our pants? In *Wearable Computers, The Fourth International Symposium on* (2000), 77–83.

[222] Villar, N., Scott, J., and Hodges, S. Prototyping with microsoft .net gadgeteer. *Tei* (2011), 377.

[223] Voida, S., Matthews, M., Abdullah, S., Xi, M. C., Green, M., Jang, W. J., Hu, D., Weinrich, J., Patil, P., Rabbi, M., et al. Moodrhythm: tracking and supporting daily rhythms. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, ACM (2013), 67–70.

[224] Vonnegut, B. A Vortex Whistle. *The Journal of the Acoustical Society of America 26*, 1 (1954), 18–20.

[225] Wadhawan, T., Lancaster, K., and Zouridakis, G. Implementation of the 7-point checklist for melanoma detection on smart handheld devices. *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (aug 2011), 3180–3183.

[226] Walters, J. A. E., Wood-Baker, R., Walls, J., and Johns, D. P. Stability of the EasyOne ultrasonic spirometer for use in general practice. *Respirology 11*, 3 (2006), 306–310.

[227] Wang, E. J., Li, W., Hawkins, D., Gernsheimer, T., Norby-slycord, C., and Patel, S. N. HemaApp : Noninvasive Blood Screening of Hemoglobin using Smartphone Cameras. In *UbiComp 2016* (2016).

[228] Wang, L., Pedersen, P. C., Strong, D., Tulu, B., and Agu, E. Wound image analysis system for diabetics. *International Society for Optics and Photonics (SPIE) 8669* (2013), 866924.

[229] Watanabe, K., and Sato, H. Vortex Whistle as a Flow Meter. In *Proc. Advanced Technologies in Instrumentation and Measurement* (1994), 1225–1228.

[230] Weberg, L., Brange, T., and Hansson, A. W. A piece of butter on the PDA display. In *Proc. CHI EA 2001*, CHI EA '01, ACM (New York, NY, USA, 2001), 435–436.

[231] Wilson, A. D., and Sarin, R. BlueTable : Connecting Wireless Mobile Devices on Interactive Surfaces Using Vision-Based Handshaking. In *Proc. GI'07* (2007).

[232] Wimmer, R. FlyEye: Grasp-Sensitive Surfaces Using Optical Fiber. *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction - TEI '10* (2010), 245–248.

[233] Wobbrock, J. O. The Future of Mobile Device Research in HCI. *Access*, Workshop on "What is the Next Generation of Human-Computer Interaction?" (2006), 131–134.

[234] Wobbrock, J. O., Findlater, L., Gergle, D., and Higgins, J. J. The aligned rank transform for nonparametric factorial analyses using only ANOVA procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2011, 143–146.

[235] Wobbrock, J. O., Kane, S. K., Gajos, K. Z., Harada, S., and Froehlich, J. Ability-Based Design: Concept, Principles and Examples. *ACM Trans. Access. Comput. 3*, 3 (apr 2011), 9:1—-9:27.

[236] Wobbrock, J. O., and Myers, B. a. Analyzing the input stream for character- level errors in unconstrained text entry evaluations. *ACM Transactions on Computer-Human Interaction 13*, 4 (2006), 458–489.

[237] Wobbrock, J. O., Myers, B. A., and Aung, H. H. The performance of hand postures in front- and back-of-device interaction for mobile computing. *International Journal of Human-Computer Studies 66*, 12 (2008), 857–875.

[238] World Health Organization. Chronic obstructed pulmonary diseases (COPD), 2015.

[239] Wu, H.-Y., Rubinstein, M., Shih, E., Guttag, J., Durand, F., and Freeman, W. T. Eulerian video magnification for revealing subtle changes in the world. *ACM Trans. Graph. (Proceedings SIGGRAPH 2012) 31*, 4 (2012).

[240] Wyatt, D., Choudhury, T., and Bilmes, J. A. Conversation detection and speaker segmentation in privacy-sensitive situated speech data. In *INTERSPEECH* (2007), 586–589.

[241] Xu, W., Huang, M.-C., Liu, J. J., Ren, F., Shen, X., Liu, X., and Sarrafzadeh, M. mCOPD: Mobile Phone Based Lung Function Diagnosis and Exercise System for COPD. *Proceedings of the 6th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA)* (2013), 45.

[242] Yach, D., Hawkes, C., Gould, C. L., and Hofman, K. J. The global burden of chronic diseases: overcoming impediments to prevention and control. *Journal of the American Medical Association 291*, 21 (2004), 2616–2622.

[243] Yamabe, T., Takagi, A., and Nakajima, T. Citron: A context information acquisition framework for personal devices. *Proceedings - 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications* (2005), 489–495.

[244] Yamabe, T., and Takahashi, K. Experiments in mobile user interface adaptation for walking users. In *Proceedings The 2007 International Conference on Intelligent Pervasive Computing, IPC 2007* (2007), 280–284.

[245] Yap, J. H., Noh, Y.-H., and Jeong, D.-U. Implementation of mobile healthcare monitoring system with portable base station, 2011.

[246] Yatani, K., and Truong, K. N. An evaluation of stylus-based text entry methods on handheld devices in stationary and mobile settings. *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services - MobileHCI '07* (2007), 487–494.

[247] Yatani, K., and Truong, K. N. BodyScope : A Wearable Acoustic Sensor for Activity Recognition. *Ubicomp'12* (2012), 341–350.

[248] Yesilada, Y., Harper, S., Chen, T., and Trewin, S. Small-device users situationally impaired by input. *Computers in Human Behavior 26*, 3 (2010), 427–435.

[249] Zhuang, Z., Kim, K.-H. K., and Singh, J. J. P. Improving energy efficiency of location sensing on smartphones. *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10* (2010), 315.

[250] Zou, H., and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology 67*, 2 (2005), 301–320.