UNOBTRUSIVE PHYSIOLOGICAL MONITORING USING SMARTPHONES

By

Tian Hao

A DISSERTATION

Submitted
to Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2015

ABSTRACT

UNOBTRUSIVE PHYSIOLOGICAL MONITORING USING SMARTPHONES

By

Tian Hao

In the past decade, we have witnessed unprecedented penetration of smartphone into our daily lives. As a personal device that is usually carried around by users, smartphone is truly the ideal platform for people-centric sensing. Together, its rich set of built-in sensors and the ever-growing computational resources enables a wealth of potential applications far beyond making calls and sending text messages. For instance, recently, Apple just launched "ResearchKit", a framework that enables iOS apps to become a powerful data collection tool for medical research.

This thesis presents an in-depth investigation in unobtrusive smartphone-based physiological monitoring, which aims to help people get healthier and fitter in a more efficient and less costly way. We have developed innovative sensing algorithms, built award-winning mobile apps, and conducted comprehensive real-world experiments for performance evaluation. The benefit of smartphone-based physiological monitoring is two-fold.

First, it enables convenient long-term monitoring, which is particularly valuable for people with chronic diseases or problems. This is mainly because the monitoring result fills in the gaps between clinic/hospital visits, and therefore, is not only able to help users better understand their condition, but also able to provide doctors with a more comprehensive view of the patient's situation. For example, we proposed *iSleep*, a contact-free and gadget-free sleep monitoring system using smartphones, which reduces users' burden to a minimum – users only need to open the app and leave the phone somewhere close to the bed (e.g., on the night stand) like they normally do. As the first sound-based sleep monitor, it helps user keep track of their sleep quality, as well as other sleep-related events such as snoring and coughing, which might be the indicator of other diseases (e.g., sleep apnea). We have evaluated iSleep based on the experiment that involves 7 participants and total 51 nights of sleep, as well the data collected from real iSleep app users. Our results

show that iSleep achieves consistently above 90% accuracy for event classification in a variety of different settings.

Second, the smartphone-based monitoring significantly increases the accessibility of health and wellness information that used to require specialized devices. For example, we presented a fitness mobile app, *RunBuddy*, which allows users to keep track of their running rhythm – the coordination between breathing and strides. As an essential part of exercise, breathing monitoring has been largely limited due to its need of customized devices, and therefore is only accessible to professional athletes and patients with certain disorders. RunBuddy is designed to be a convenient and unobtrusive exercise feedback system, and only utilizes commodity devices including smartphone and Bluetooth headset. By employing a physiological model, RunBuddy is able to provide a reliable running rhythm measurement, which is more intuitive than breathing rate, and also considered as a good indicator of the user's fitness level. Our extensive evaluation involving 13 subjects and 39 runs suggests that RunBuddy can correctly measure the running rhythm for indoor/outdoor running 92.7% of the time.

To achieve a through and comprehensive performance evaluation, we have validated our systems using data collected from real users in real-world scenarios. Our data collection procedures have been reviewed and approved by the Institutional Review Board (IRB) at Michigan State University (IRB#12-1178 for iSleep and IRB#13-791 for RunBuddy).

ACKNOWLEDGEMENTS

First of all, I would like to thank my adviser Dr. Guoliang Xing for his great support and candid opinions. With his guidance and influence, I have thrived from an undergraduate student to a researcher, who is able to identify research problems, generate innovative ideas, and produce high-quality research. Thank you for always being patient, and giving me the freedom and valuable advices on research for the past several years. Also, I feel fortunate to have had the opportunity to collaborate with Dr. Gang Zhou in College of William and Mary who has added significant value to my work.

I would also like to thank other members in Dr. Xing's group. I feel lucky to work with them and exchange ideas in the group meeting. To me, they are not only outstanding researchers, but also good friends.

Finally, I can not thank enough my family and friends for their unconditional love and support. I must express my tremendous appreciation to my families, especially my mother and father, for supporting me in pursuing my dream halfway across the world. I do not think I could have made my Ph.D. life without their love, understanding and encouragement. Thank you for everything.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

The era of smartphones has dramatically overthrown our impressions to what phones are capable of. Unlike feature phones, smartphones are not any more limited to be used for phone calls and text messages. Rather, they are considered as the first truly mobile computing device with sensing and internet capabilities. We use it everyday to check emails, search places for dinner, or even estimate our daily energy expenditure. Most of these intriguing applications would not be possible without a rich set of built-in sensors in smartphone, such as accelerometer, GPS and microphone. Along with its increasingly powerful computational resource, these sensors bring smartphones even more central to people's everyday lives by enabling many new emerging application domains like mobile health and well-being applications. An example of such mobile applications is sleep quality monitoring, which enables users to track how well they sleep over night just by using smartphones. Smartphone applications of this kind, such as Sleep as Android [3] and Sleep Cycle [5] have been very popular and had millions of downloads. Their unsurprising popularity is mostly because of the fact that they succeed to provide a low-cost and, more importantly, convenient way for nearly everyone to be aware of their sleep quality, as long as they have a smartphone.

1.1.1 Smartphone-based Physiological Monitoring

Physiological monitoring like sleep quality monitoring plays a key role in the in diagnosis and early discovery of many disorders and diseases. Traditonally, the monitoring involves numerous sensors and complex instruments, and is thus limited in the clinic or hospital settings. Thanks to the smartphone-based mobile health apps, the bar of monitoring and analyzing one's physiological events has been significantly lowered. What used to be possible only in clinics or hospitals can now

be done everyday and anywhere. Moreover, through analyzing the detected events, the application is also able to provide a personalized feedback to users, and assist them in making positive changes to achieve a healthier lifestyle.

Take the application of sleep quality monitoring for an example. The quality of sleep is an important factor in maintaining a healthy lifestyle. By analyzing long-term data about sleep pattern, users are able to keep track of their sleep quality, discover possible sleep disorders, and make positive changes to their lifestyles. However, until now, technology has not enabled personalized, in-place sleep quality monitoring and analysis. As the primary clinical tool for sleep monitoring, Polysomnography (PSG) [23] can provide a quantitative profiling of sleep to diagnose sleep disorders. However, due to the need of various sensors, PSG-based sleep quality measurement is usually limited to clinical settings and found very invasive by most patients. Hence, smartphone-based sleep quality monitoring offers users a much more convenient and "hospital-free" way of tracking their sleep quality, which allows more people to be aware of their sleep quality.

Another example of physiological monitoring would be running rhythm monitoring. As one of the most popular exercises, running is accomplished through a tight cooperation between the respiratory and locomotor systems, modulated by both mechanical and neurological interactions. Research has suggested that a proper *running rhythm* – the coordination between breathing and strides – helps to improve exercise efficiency and postpone fatigue. The ability of maintaining a stable running rhythm is also considered as a good indicator of the runner's fitness level. Unfortunately, to date, there has been no convenient and unobtrusive way of measuring running rhythm continuously. Cardiopulmonary exercise testing (CPET) is a widely adopted clinical tool to evaluate exercise capacity. It provides an analysis of respiratory gas exchange and cardiac function during exercise. However, CPET is usually limited to hospitals and clinics, due to its complicated procedure and high cost (about $\$20,000$ /unit). A smartphone-based running rhythm monitoring system would allow more users to improve their exercise experience by keeping track of their breathing pattern on a daily basis.

However, it is not trivial to design such mobile health applications for physiological monitor-

ing, because of the following requirements. First, it needs to be unobtrusive. It should minimize the burden on the user, and the user should not feel any kind of uncomfort when using the system. Second, it should manage to take full advantage of various built-in sensors, and provide fine-grained physiological measurement (e.g., sleep status). Third, the performance of the system needs to be reliable across different users, environments and smartphone platforms. Fourth, The users' privacy needs to be strictly protected. Due to the inherently private nature of of physiological events (e.g., sleep), any concern (or even suspension) of privacy breach may prevent the adoption of such applications. For example, in the case of sleep quality monitoring, the system should process the sensor samples on the fly and only keep sleep-related data (e.g., the number/loudness of snores), instead of sending any raw sensor samples to a remote server, because they may capture sensitive information such as audio of sleep talks, conversations before/after sleep and etc. Lastly, as a supplementary service, the application shouldn't drain a large proportion of the phone's battery. Therefore, the design should also consider the power consumption of the application, especially for the applications that require continuous sensing.

1.1.2 Problem Statement

Having introduced the field of smartphone-based physiological sensing, now we discuss the key challenges that we need to deal with in developing a smartphone-based system for physiological monitoring.

First, the system needs to effectively monitor the physiological events in an unobtrusive manner. Typically, the events can be detected by sampling and analyzing the data collected from the built-in sensors such as microphone or accelerometer. However, the built-in sensor of smartphone usually has low sensitivity and is more likely to be interfered by environmental noises. For example, the built-in microphone of smartphone is designed for capturing close vocals, and usually has low sensitivity. Therefore, the captured acoustic signal associated with the interested event typically may not only be incomplete, but also contain considerable amount of environmental noises.

Second, the system needs to be able to provide robust detection result across different users

and environments. The inherent nature of physiological events determines that people act different from each other even for the same kind of event. For instance, in the case of sound-based snoring detection, different people likely snore in different ways in terms of sound frequency and loudness. Moreover, even the same person may snore differently due to the change of body position overnight. Another important issue to consider is that, the system should also be able to detect events in a robust manner with the interference of different environmental noises. This is especially challenging when the system is used in a changing environment. Again, take the snore detection for an example, the sound profile of the environmental noise could be constantly changing, because of the operating appliances such as fans and air conditioner.

Lastly, in order to preserve users' privacy, the system can not store or transmit raw signals. Instead, the captured signal should be processed locally on the smartphone in real-time, while only the event detection results are kept and shown to the user. This is very challenging when the signal is sampled at a high rate. In such situation, due to the resource constrains of smartphones, the processing algorithms must be extremely lightweight in order to process the data in real-time, while maintaining satisfactory event detection accuracy.

1.2 Outline

In addressing these aforementioned challenges, we have developed innovative sensing algorithms, built award-winning prototype systems, and conducted comprehensive real-world experiments for performance evaluation. The following is an outline of this report.

1.2.1 Smartphone-based Unobtrusive Physiological Monitoring

In Chapter 3, we begin our investigation of unobtrusive smartphone-based physiological sensing by targeting the application of sleep quality monitoring. Keeping all the aforementioned requirements in mind, we have carefully designed the application to make it truly unobtrusive, while still delivering satisfactory detection results. The quality of sleep is an important factor in maintain-

ing a healthy life style, yet current sleep monitoring systems are often difficult to use and hence limited to sleep clinics, or invasive to users, e.g., requiring users to wear a device during sleep. Therefore, we have designed iSleep – a practical system to monitor an individual's sleep quality using off-the-shelf smartphone. iSleep uses the built-in microphone of the smartphone to detect the events that are closely related to sleep quality, including body movement, couch and snore, and infers quantitative measures of sleep quality. iSleep adopts a lightweight decision-tree-based algorithm to classify various events based on carefully selected acoustic features, and tracks the dynamic ambient noise characteristics to improve the robustness of classification. We have evaluated iSleep based on the experiment that involves 7 participants and total 51 nights of sleep, as well the data collected from real iSleep users. Our results show that iSleep achieves consistently above 90% accuracy for event classification in a variety of different settings. By providing a fine-grained sleep profile that depicts details of sleep-related events, iSleep allows the user to track the sleep efficiency over time and relate irregular sleep patterns to possible causes.

1.2.2 Leveraging Physiological Model to Improve Sensing Accuracy

In Chapter 4, we propose a novel idea of exploiting physiological model to improve the accuracy of smartphone-based physiological sensing. We demonstrate the idea by proposing the design of RunBuddy – the first smartphone-based system for continuous running rhythm monitoring. As one of the most popular exercises, running is accomplished through a tight cooperation between the respiratory and locomotor systems, modulated by both mechanical and neurological interactions. Research has suggested that a proper *running rhythm* – the coordination between breathing and strides – helps to improve exercise efficiency and postpone fatigue. The ability of maintaining a stable running rhythm is also considered as a good indicator of the runner's fitness level. RunBuddy is designed to be a convenient and unobtrusive exercise feedback system, and only utilizes commodity devices including smartphone and Bluetooth headset. We propose a novel approach that integrates ambient sensing based on accelerometer and microphone, and a physiological model called Locomotor Respiratory Coupling (LRC), which indicates possible ratios

between the stride and breathing frequencies. A key challenge in the design of RunBuddy is that the sound of breathing typically has very low intensity and is susceptible to interference. RunBuddy adopts lightweight signal processing pipelines to detect breaths and strides, and correlate possible LRC ratios, strides, and the acoustic breath detection results together to obtain reliable running rhythm measurement. We evaluate RunBuddy through extensive experiments involving 13 subjects and 39 runs. Our results show that, by leveraging the LRC model, RunBuddy correctly measures the running rhythm for indoor/outdoor running 92.7% of the time. Moreover, RunBuddy also provides detailed physiological profile of running (e.g., the stability of running rhythm – an indicator of fitness level) that can help users better understand their running process and improve exercise self-efficacy.

CHAPTER 2

BACKGROUND

This chapter aims to present the progress made thus far in the field of physiological monitoring by mainly focusing on two specific areas, which are sleep quality assessment and exercise monitoring systems. For both areas, we begin by discussing the traditional approaches, followed by the efforts that have been made to enable in-place and clinical-free monitoring. Lastly, we briefly present the current progress in acoustic mobile sensing.

2.1 Sleep Quality Assessment

According to AASM (American Academy of Sleep Medicine), the sleep stage scoring based on polysomnography (PSG) has long been considered as the “gold standard" of sleep study [36]. A polysomnogram typically requires the recording of multiple channels including electroencephalography (EEG), electromyography (EMG), electrocardiography (ECG) or heart rate, respiratory effort, air flow, oxygen saturation and etc. [23]. The result of PSG includes a collection of indices such as sleep onset latency, total sleep time and etc, which are considered together to infer the sleep quality. Due to the need of various sensors, PSG-based sleep quality measurement is usually limited to sleep clinics.

Actigraphy has been studied as an inexpensive alternative to assess human sleep and wakefulness [11] based on the subject’s body movements overnight. The basic idea is that the state of sleep and wake can be inferred from the amount of body movement during sleep [11]. Through processing the logged acceleration data, epoch-by-epoch (usually 30 second or 1 minute) sleep/wake predictions are calculated. Several algorithms [33, 68, 26] have been proposed to derive sleep quality from actigraphy. The average accuracy of predicting sleep/wake state is around 90% (reported 88% in [26] and 94-96% in [75]).

A widely used subjective sleep quality assessment method is through PSQI (Pittsburgh Sleep

Quality Index) [22], which is a self-rated questionnaire to assess the sleep quality and disturbance over a long-term interval. In PSQI, a set of sleep measures are collected, including sleep latency, sleep duration, sleep disturbance and etc. PSQI has been shown useful in numerous studies [13] [21] over a variety of populations. However, the accuracy of PSQI is highly variable and is often impeded by the inaccuracy of subject's memory and perception.

Several commercial personal sleep assessment products are currently available. Watch PAT [8] detects respiratory disturbances during sleep by monitoring peripheral arterial tone (PAT). The users are required to attach a probe to their finger during sleep. ZEO [9] is a popular sleep monitoring product that infers sleep stages using three EEG sensors contained in a head band worn by the user during sleep. Several actigraphy-based products such as Sleep Tracker [7] and fitbit [1] require the user to wear the device containing accelerometer during sleep.

Recently, several research efforts aimed at developing low-cost sleep assessment systems. In [67], a wearable neck-cuff system for real-time sleep monitoring is designed based on oximetry sensor, microphone and accelerometer. Instead of directly measuring the sleep, SleepMiner [14] predicts the sleep quality based on the user's daily context information such as sound, light, postures, and positions. In [35], the body position and movements during sleep are monitored using accelerometers attached to bed mattress. A dense pressure sensitive bedsheet for sleep posture monitoring is proposed in [47]. However, these systems incur nontrivial monetary costs of hardware or professional installation.

Several Android and iOS Apps such as *Sleep as Android* [4] and *Sleep Cycle* [6] can measure sleep quality. All of them exclusively rely on the actigraphy-based methods that monitor body movements overnight using smartphones. However, sleep-related events such as cough and snore can not be reliably detected based on acceleration. For example, snore is the sound caused by the vibration of respiratory structures while sleeping due to obstructed air movement, and is not necessarily associated with body motion. Moreover, since the motion data is collected through the built-in accelerometer, the phone must be put on the bed, which not only is inconsistent with the habit of most users, but also may obstruct the individual's body movement.

2.2 Exercise Monitoring Systems

Recently, cardiopulmonary exercise testing (CPET) has become an important clinical tool to evaluate exercise capacity and predict outcome in patients with various cardiac conditions. It has been increasingly used in exercise training. It provides a breath-by-breath analysis of respiratory gas exchange and cardiac function at rest and during a period of exercise. The fine-grained measurement relies on various sensors, such as electrocardiographic (ECG) sensor for heart monitoring, and a spirometer connected using a mouthpiece for breath collection. However, CPET is usually limited to hospitals and clinics, due to its complicated procedure and high costs (about $\$20,000$ /unit).

Several wireless commercial CPET products are available on the market. For example, Oxycon Mobile [62] integrates various sensors along with batteries into a vest, which is worn by the subject during the test. However, it is mainly designed to evaluate the training of professional athletes, and not suitable for everyday use due to the bulky size and high cost.

Several efforts have been made to develop smartphone-based exercise monitoring systems. For example, a system for monitoring the workload (a variation of heart rate) is proposed in [71]. It leverages machine learning techniques to predict heart rate variation from the acceleration and speed during walking. In [76], the authors present a smartphone-based approach to estimate caloric expenditure of bicyclists. They improve the accuracy by considering multiple inputs such as GPS signal and map information. SpiroSmart, a smartphone-based spirometer using the built-in microphone is proposed in [43]. However, in order to measure air flow and volume of a single breath, the user has to blow air to the phone's microphone in close proximity in a quiet environment. Therefore, it is not suitable for continuous breath monitoring during exercise.

A few methods have been proposed to detect strides based on acceleration. In [19], common stride detection algorithms are evaluated based on large amount of data from smartphone sensors. A gait analyzer is proposed in [38] that can identify gaits such as walking and running solely using acceleration. Reliable stride detection has also been used in many other applications, such as a meter-level indoor positioning system [46] and a system that automatically monitors energy expenditure [45].

In addition to the exercise monitoring systems mentioned above, several systems have been developed to keep users motivated and engaged during exercise, by providing real-time information about the user's performance. For example, by continuously monitoring the user's heart rate or activity level, the smartphone may provide music suggestions or feedbacks that guide users' steps [27, 17] or help users achieve their target heart rate [60] during exercise.

2.3 Acoustic Mobile Sensing

Recently, several systems have been developed to detect events of interests using the microphone on smartphones. In [44], a privacy-preserving approach for detecting cough using smartphones is proposed. StressSense [50] was proposed to recognize stress from human voice using smartphones. SoundSense [49] adopts a light-weight and scalable architecture to recognize different kinds of ambient sounds such as speech and music. In [24], a location classification system based on captured images and sound is presented, where a GMM-based sound classifier is trained to detect acoustic events such as car and crowd noise.

CHAPTER 3

SMARTPHONE-BASED UNOBTRUSIVE SLEEP MONITORING

3.1 Introduction

Maintaining a health lifestyle is one of the key factor that impacts our personal health. For example, the daily life choices we make such as sleep and exercise are deeply connected to a wide range of health-related problems including high-blood pressure, stress [61], anxiety, diabetes and depression [28, 31]. The current smartphones equipped with a wide variety of sensors spare us from manual data entry, and makes it possible for us to keep track of our everyday behaviors and further assist us in maintaining a healthy lifestyle through feedbacks. However, there are still a number of technical barriers need to be tackled to make these personal wellbeing applications truly benefit the end uses. In this chapter, we demonstrate the feasibility of smartphone-based physiological sensing by focusing on a specific health dimension, which is sleep. We present the design, implementation and evaluation of iSleep – an unobtrusive, portable tool for in-place sleep monitoring using smartphone. Throughout the development of iSleep, we have identified and solved a number of key challenges of mobile sensing applications.

3.2 Overview

iSleep is designed to be a light-weight sleep quality monitoring system that is reliable in detecting sleep-related events across different users and environments. Fig. 3.1 shows the architecture of the iSleep system. First, the acoustic signal is continuously sampled at the frequency of 16 *kHz* from the microphone, and segmented into *frames*. Second, the acoustic frames are fed to *Noise Detection*, where the system determines whether a frame only contains the sound of ambient noise. The model of ambient noise is then updated based on detected noise frames. As a result, the system is able to adapt to the changes of ambient noise. Third, acoustic features such as root

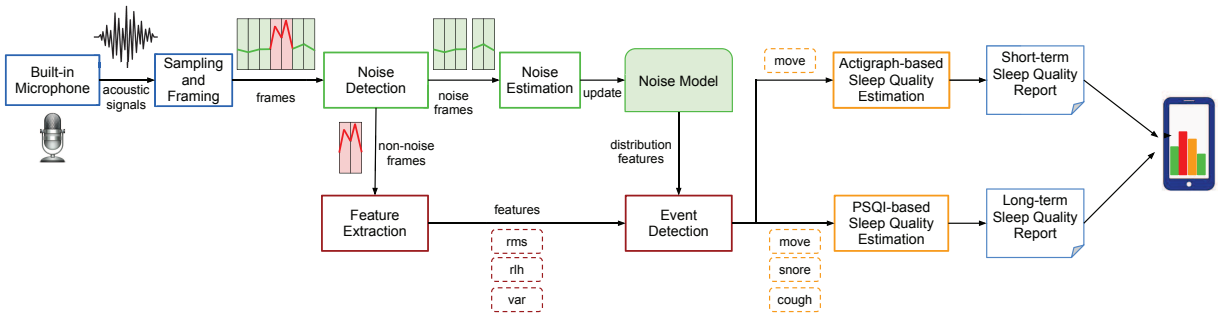


Figure 3.1 The architecture of iSleep system.

mean square and variance will be extracted from the frames that potentially contain events of interest. The extracted features, along with the updated ambient noise model, are fed to the *Sleep Event Detection*, where sleep-related events such as snoring, coughing and body movement will be detected.

iSleep derives both short-term (one-night) and long-term sleep quality from sleep-related events according to two well-established sleep scoring criteria: actigraphy and Pittsburgh Sleep Quality Index (PSQI) [22]. iSleep uses actigraphy to estimate the sleep/wake states overnight and then computes a metric called *sleep efficiency*, which is the ratio of actual sleep time to total in-bed time. Compared with other quality measures such as sleep stages, sleep efficiency provides a quantitative and more intuitive feedback to users. In addition to one-night sleep efficiency, iSleep employs PSQI to estimate long-term sleep quality over multiple nights. PSQI is a self-rated questionnaire which assesses sleep quality and disturbances over a long time interval. Based on the detected events such as snoring and coughing, iSleep is able to estimate the answers to several PSQI questions such as “During the past month, how often have you had trouble sleeping because you cough or snore loudly?". Then a sleep quality score can be calculated based on scoring rules specified by PSQI.

As an unobtrusive, portable tool for in-place sleep monitoring, Sleep has a potential in helping users improve their sleep quality and stay healthy in many ways. For example, by providing a sleeping profile that depicts details of sleep-related events, iSleep allows the user to track the sleep

efficiency over time, relate bad sleep to possible causes like extensive snores which are otherwise hard to identify, and help their healthcare providers diagnose trends related to certain diseases. Moreover, the fine-grained sleep events detected by iSleep can greatly improve the fidelity of subjective, questionnaire-based sleep assessment tools like PSQI whose utility is otherwise impeded by the inaccuracy of subject's memory and perception.

3.3 System Requirement and Challenges

iSleep is designed to be a "sleep diary" that provides the user real-time, fine-grained feedback to their sleep quality on a daily basis [1]. Specifically, iSleep is designed to meet the following requirements: (1) Since iSleep operates over night while the user is asleep, it needs to be unobtrusive. It should minimize the burden on the user, and the user should not feel any kind of uncomfort when using the system. (2) iSleep needs to provide fine-grained measurement, such as overall sleep efficiency and the occurrences of events that may interrupt sleep, such as cough and snore. Such fine-grained sleep profiling helps the user understand what factors affect their sleep quality. (3) iSleep needs to deliver robust monitoring accuracy across different users, smartphones and sleep environments. (4) The users' privacy needs to be strictly protected. Due to the inherently private nature of sleep, any concern (or even suspension) of privacy breach may prevent the adoption of sleep monitoring technology like iSleep. For instance, the system should process the sensor samples on the fly and only keep sleep-related data (e.g., the number/loudness of snores), instead of sending any raw sensor samples to a remote server, because they may capture sensitive information such as audio of sleep talks, conversations before/after sleep and etc.

To meet these requirements, three major challenges need to be addressed in developing iSleep. First, in order to effectively monitor the sleep quality in an unobtrusive manner, iSleep samples and analyzes acoustic signals from the built-in microphone to detect sleep-related events. Therefore,

[1] iSleep is not designed or certified for clinical use, although the monitoring results provided by iSleep could be potentially useful for professional diagnosis of sleep-related disease such as insomnia [11].

the user only needs to leave the phone somewhere close to the bed (up to several meters). However, the built-in microphone of smartphone is designed for capturing close vocals, and usually has low sensitivity. Moreover, many sleep-related events only generate low-intensity sound. For example, the intensity of the sound from a roll-over movement is typically only several *dB* higher than that of ambient noise.

Second, iSleep needs to detect sleep-related events in a robust manner across different users and environments. For instance, different people likely snore in different ways in terms of sound frequency and loudness. Moreover, even the same person may snore differently due to the change of body position overnight. Noises from appliances such as fans may also have a major impact on the acoustic event detection accuracy.

Lastly, in order to preserve users' privacy, iSleep does not store or transmit raw sound samples. Instead, sound data is processed locally on the smartphone in real-time, while only the event detection results such as the number of occurrences of snore/cough/body movement are kept and shown to the user. To capture the features of various acoustic events, the microphone must be sampled at a high rate. Due to the resource constrains of smartphones, the acoustic processing algorithms must be extremely lightweight in order to process the data in real-time, while maintaining satisfactory event detection accuracy.

3.4 System Design

In this section, we describe the design of iSleep. First, we discuss the sleep-related events that iSleep can detect, and the acoustic features used to detect those events. Next, we describe how to estimate ambient noise. Lastly, we discuss sleep-related event classification. Our design is based on careful analysis of real data of a long-term experiment that involves 7 subjects and total 51 nights of sleep.

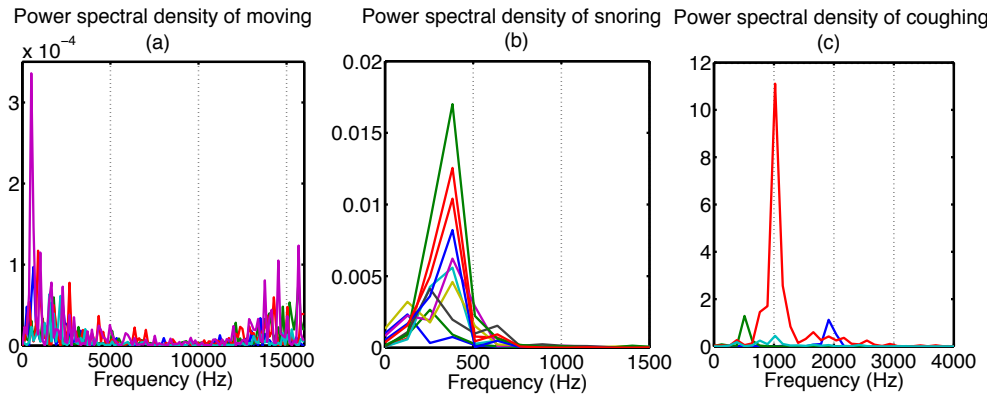


Figure 3.2 (a), (b) and (c) show the power spectral density of typical moving, snoring and coughing events, respectively.

3.4.1 Sleep Events and Feature Extraction

Since most people sleep in a relatively quiet environment at night, iSleep categorizes the possible sounds during sleep into sleep-related events, ambient noise, and other sounds such as those caused by cars/trains passing by. Specifically, the sleep-related events of interest include body movement, snoring and coughing. Our key insight is that, even though the acoustic profiles of sleep events are highly dependent on each individual, they have distinguishable features in terms of energy and frequency. For example, the dominant frequency of snoring is much lower than that of other events. iSleep requires user to manually start and close the app when he goes to bed and gets up, respectively. The duration of sleep is hence equal to the running time of iSleep. We note that it is possible to infer the time that the user goes to bed or gets up by combining accelerometer microphone readings. However, this requires the smartphone to constantly sample sensors, resulting in high energy consumption.

In order to build a light-weight classifier that can adapt to different individuals and environments, we choose three features based on the key characteristics of each sleep-related event. The first feature is root mean square (*rms*), which captures the loudness of sound. Let f be a frame that consists of n samples of acoustic amplitude $s_1, s_2, ..., s_n$. In our implementation, each frame contains $1,600$ acoustic samples collected at 16 *kHz*. The *rms* of the frame is given by

$$rms(f) = \sqrt{\frac{s_1^2 + s_2^2 + ... + s_n^2}{n}} \tag{3.1}$$

where $rms(f)$ denotes the value of *rms* for frame f. Fig. 3.2 shows the power spectrals of moving, snoring and coughing. We can see that their energy distributions are different. For example, most energy of snoring concentrates on low-frequency band ($0 \sim 750Hz$), while the energy of moving distributes on both high and low frequency bands. Based on this observation, the second feature we choose is the ratio of low-band to high-band energies (*rlh*). The change of *rlh* over time reflects the transition of dominant frequency. The rise of *rlh* indicates that the proportion of low-band energy in the total energy is increasing. In other words, the dominant frequency is transiting in the direction of high to low. For example, the dominant frequency of snoring is significantly lower than that of the ambient noise and other activities. As a result, the rise of *rlh* can be used to effectively detect snoring. In order to compute the *rlh* of frame f, we need to calculate the energy of frame f in both low and high frequency bands. The low-band frame f^l is calculated by applying a low-pass filter as follows,

$$s_i^l = s_{i-1}^l + \alpha \times (s_i - s_{i-1}^l) \tag{3.2}$$

where s_i^l is the i-th acoustic sample of the low-band frame f^l, and the default value of α is set to be 0.25. The high-band frame f^h is computed by applying a high-pass filter as follows,

$$s_i^h = \alpha \times (s_{i-1}^h + s_i - s_{i-1}) \tag{3.3}$$

where s_i^h is the i-th acoustic sample of the high-band frame f^h, and the default value of α is set to be 0.25. Then the *rlh* of frame f, $rlh(f)$, is given by

$$rlh(f) = \frac{rms(f^l)}{rms(f^h)} \tag{3.4}$$

Fast Fourier Transform (FFT) and Mel-frequency cepstral coefficients (MFCC) [37] are two commonly used frequency-domain features for acoustic analysis. However, compared with them, *rlh*

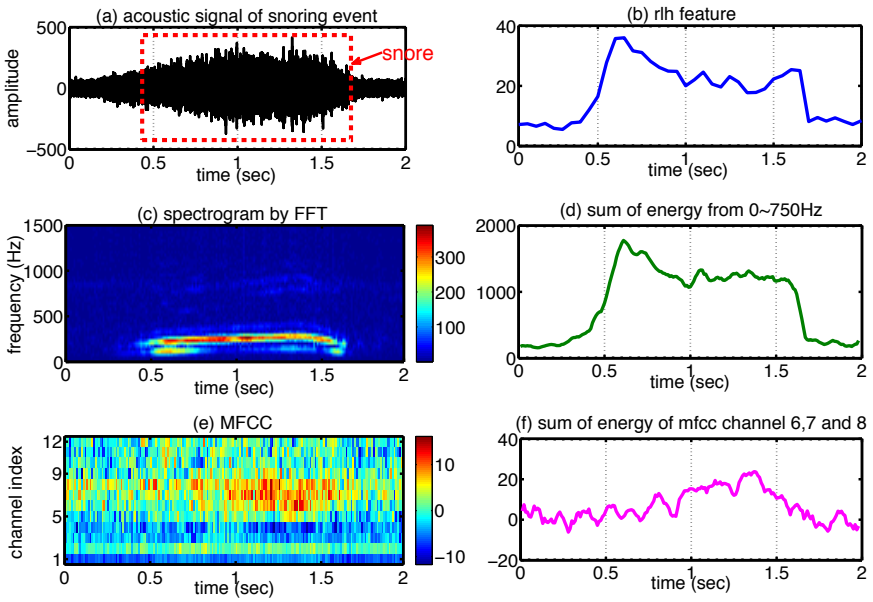


Figure 3.3 Comparison of different frequency-domain features including *rlh*, FFT and MFCC. The comparison is based on a 2-second recording containing a snoring event. (a) shows the acoustic signal of the recording. The part within the red rectangle represents the snoring event. (b) shows the *rlh* extracted from this recording. (c) and (d) show the spectrogram calculated by FFT and the summation of low-frequency (0-750Hz) energy, respectively. (e) and (f) show the energy of 12 MFCC channels and the total energy from channel 6 to 8, respectively.

has significantly lower computation overhead, while yielding the same detection performance for our application. The computation of normalized *rlh* within a frame (containing 1600 samples) requires around 11200 addition and 8000 multiplication operations, while that of FFT needs over 54000 addition and 36000 multiplication operations [29]. Fig. 3.3 shows the comparison among the three aforementioned features based on a two-second recording of snoring. We can see that, *rlh* (Fig. 3.3(b)) yields a similar result as FFT (Fig. 3.3(d)), which can be used to clearly detect the snoring. The performance of MFCC (Fig. 3.3(f)) is inferior to that of *rlh*. Designed for speech recognition, MFCC is more sensitive to vocal sound, which have very different characteristics from snoring. The third feature is variance (*var*), which reflects how far the amplitudes of acoustic signals within the frame are spread out. For example, the *var* of a frame associated with body movement is typically much lower than that of snoring or coughing.

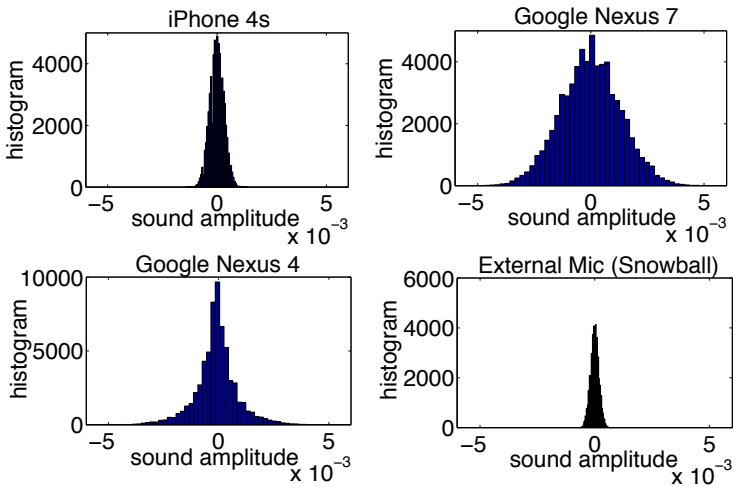


Figure 3.4 Histograms of recorder noise generated by different devices. The duration of acoustic data used is 4 seconds. The x-axis indicates the normalized amplitude.

3.4.2 Noise Characterization

In a typical scenario, the events related to sleep quality, including body movement and snoring, are rare while most sounds are noise generated by various sources. Thus noise identification is critical to the accuracy of detecting sleep-related events. We refer to the ambient sounds that last for a relatively long period as noises, as opposed to the short-duration sounds that are caused by a sleeping individual. Specifically, acoustic noise is caused by two major sources. The first is the background noise generated by the recording unit itself while recording. The level of recorder noise of built-in microphone of smartphone is much higher than that of external standalone microphone. In addition, the recorder noise level varies with different smartphones.

Fig. 3.4 shows the amplitude distributions of recorder noises collected by sampling the microphones of 4 different devices (iPhone 4s, Nexus 7, Nexus 4 and an external conference microphone) in a quiet room for 4 seconds. The sampling frequency is 16 kHz and the value of each sample is scaled to $[-1, 1]$ from its original 16-bit representation. We can observe that the noise level of external conference microphone is substantially lower than those of the built-in microphones of smartphones. Among the smartphones, iPhone 4s generates the lowest level of recorder noise.

Another common source of noise is appliances that are operating overnight (e.g., fan or air-conditioner). The mean and standard deviation across different groups differ substantially. For example, the operation of A/C results in a wider distribution in all three features (*rms*, *rlh* and *var*). However, a key observation is that most values in all three cases fall within the range of $mean \pm 3 \times std$, regardless of the types of the noises or smartphones used.

3.4.3 Noise Model Estimation

The result in Section 3.4.2 suggests that, the key feature that differentiates noise from event-related sound is its relatively stable variance. This is due to the fact that the noise does not vary substantially within a short duration (i.e., a few seconds). This observation allows us to design a simple yet robust sleep-event detection method based on the noise profile. The basic idea is to detect events based on the thresholds of features that are calculated using the noise measurement. To adapt to different environments, the system continuously detects and updates the current noise model, which is used to calculate the thresholds used to detect and classify sleep events.

Specifically, iSleep first detects noise from a sequence of frames with stable standard deviations. It involves two steps. First, the system calculates the standard deviation $std_i = std(f_i)$, where f_i denotes the i-th frame, which captures the stability of acoustic signal within a frame. However, the standard deviation varies with different devices and noise profiles. Therefore, in order to to improve the robustness of noise detection, we normalize the standard deviation of each frame within a T-second window (containing 40 frames in our implementation) as follows:

$$\overline{std_i} = \frac{std_i - std_{mean}}{std_{mean} - std_{min}} \tag{3.5}$$

where std_{mean} and std_{min} denote the mean and minimum standard deviation within the current window W. Second, the system calculates the variance of the normalized standard deviation within the window W. Fig. 3.5 shows the histogram of the variance based on $3,644$ noise windows collected from real experiments conducted by different subjects. We can see that the variances of most noise windows are grouped within [0.4,0.5]. More than 95% variances are below 0.5.

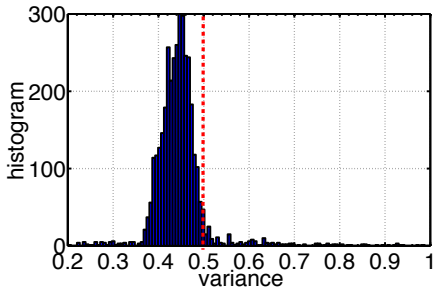


Figure 3.5 The histogram of variance of $\overline{std}$ (Equ. 3.5) within 4-second noise window. The data is extracted from real experiments, containing $3,644$ noise windows ($14,576$ seconds).

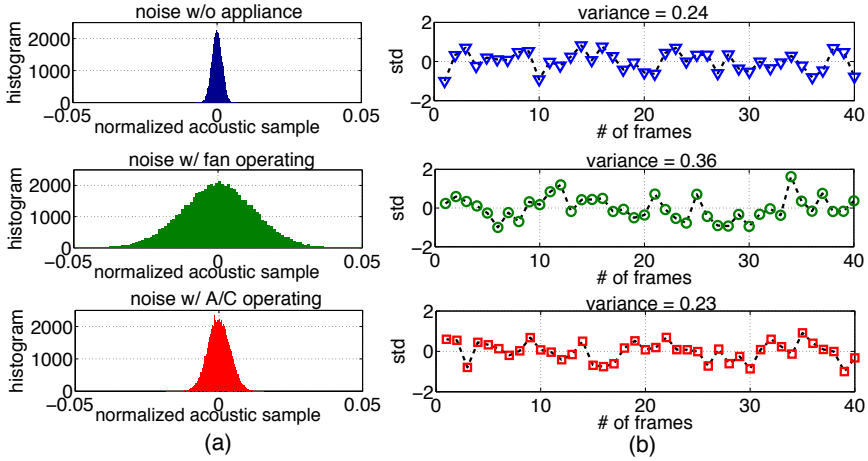


Figure 3.6 (a) The histograms of sound intensity of different types of noises. (b) The normalized standard deviation of each frame. The acoustic data is collected by iPhone 4s for a duration of 4 seconds.

Therefore, we use 0.5 as a threshold to detect noise. Specifically, the frames within a noise window will be considered as noise if the variance is lower than 0.5.

Fig. 3.6(b) plots the normalized standard deviation of different noises. We can see that, even though they have different acoustic amplitude distribution (shown in Fig. 3.6(b)), the normalized standard deviations are similar. Since their variances are lower than the preset threshold 0.5, they will be classified as noise. The histogram in Fig. 3.7 shows the distribution of acoustic signals for a duration of 4 seconds. It contains a slight body movement lasting from around $2.8s$ to $3.8s$. As the sound of the movement is very slight, its distribution is close to that of noise without operating appliance. However, we can observe that the normalized standard deviation has a variance of 1.75,

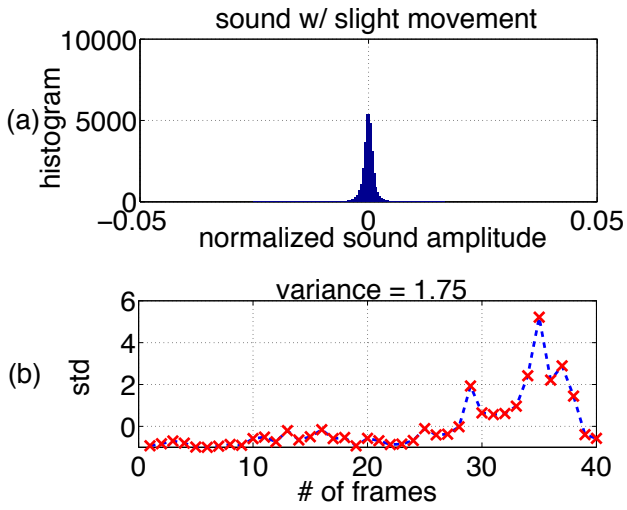


Figure 3.7 (a) The histogram of sound intensity of a 4-second recording containing body movement. (b) The normalized standard deviation of each frame. The acoustic data is collected by iPhone 4s for a duration of 4 seconds.

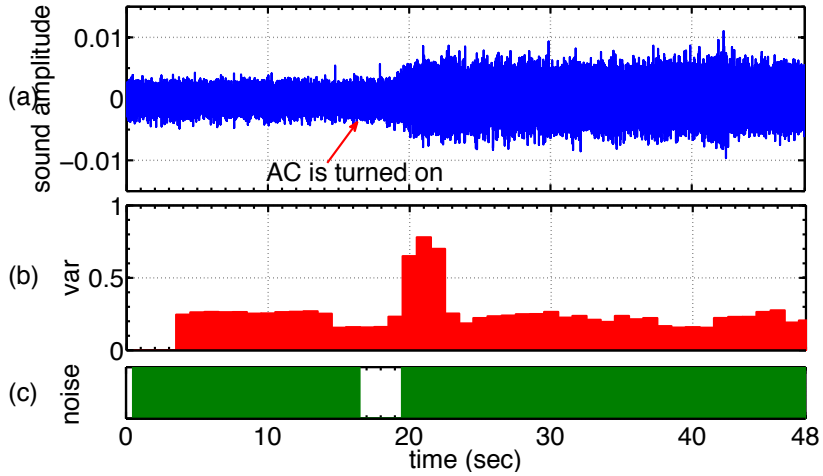


Figure 3.8 The process of detecting changing noise. (a) shows the amplitude of acoustic signals sampled in the bedroom at night. Around the 19-th second, the air conditioner was turned on. (b) shows the variance of the signal over the last 4 seconds. (c) shows the result of noise detection.

which clearly reflects the movement event. Therefore, these frames will be classified as frames of interest and fed into the feature extraction component.

A typical scenario at night is that the fan of A/C or heater is automatically turned on and off, leading to a changing noise profile. Fig. 3.8 shows the scenario where the air conditioner is turned

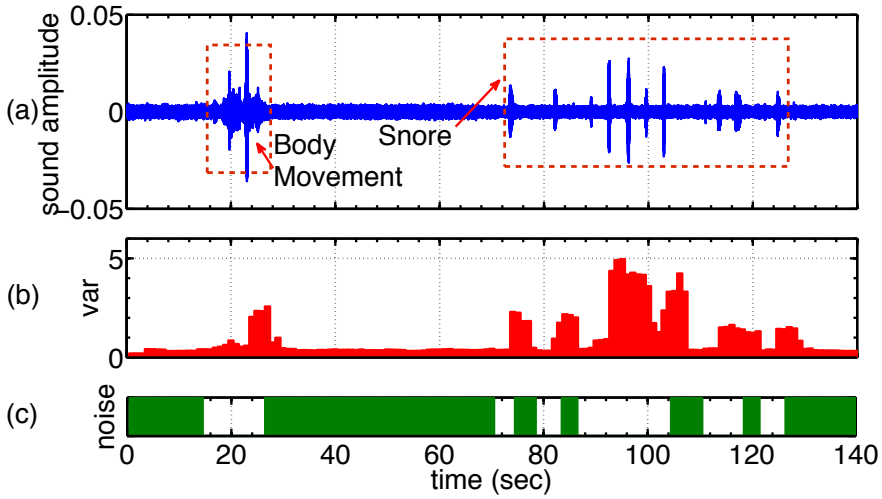


Figure 3.9 The process of detecting noise when the sounds of sleep events are included. (a) shows the sound wave of a 140-second audio collected by Nexus 4 with the sound of body movement and snoring. (b) shows the variance of the sound signal over the last 4 seconds. (c) shows the result of noise detection.

on. Fig. 3.8(b) shows the variance over the past 4 seconds. In the detection result shown in Fig. 3.8(c), we can observe that only a short period corresponding to the transition is detected as non-noise sound. As such noise misclassification only occurs in occasional transient states of appliances, it does not affect the accuracy of sleep-related event detection. Fig. 3.9 shows another typical scenario where the sounds of sleep events are included. We can see that only the parts without human generated sounds are detected as noise. Therefore, our noise detection algorithm is able to effectively detect changing noise.

After a sequence of frames is detected as noise frames, iSleep calculates three features for each of the frames. In order to estimate the current noise, iSleep computes the mean and standard deviation ($mean(rms)$, $std(rms)$, $mean(rlh)$, $std(rlh)$, $mean(var)$ and $std(var)$) for each feature. Then, each newly calculated distribution feature F_{new} will be used to update the current corresponding feature (F_{cur}) according to an Exponential Moving Average (EMA) algorithm as follows,

$$F_{cur} = F_{cur} + \beta \times (F_{new} - F_{cur}) \tag{3.6}$$

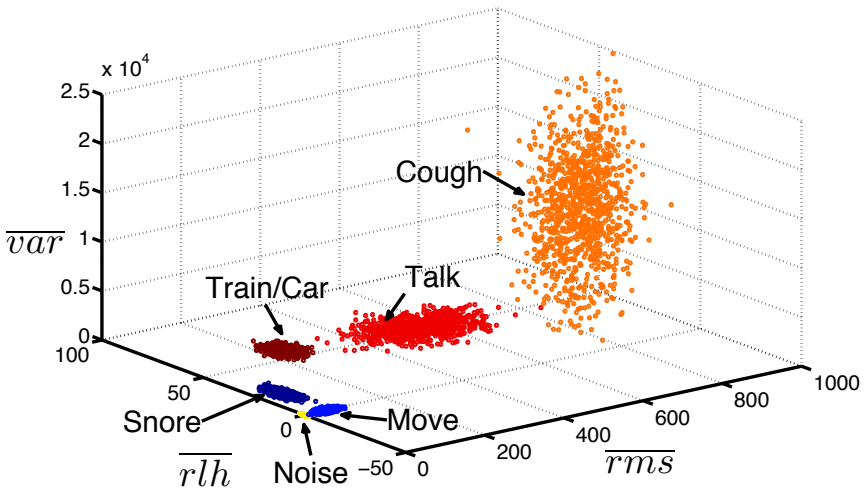


Figure 3.10 The sound feature vectors of different events in feature space.

In our implementation, the default value of β is set to be 0.5. The EMA algorithm ensures that estimated noise model adapt to the changing noise profile. After the features of current noise are updated, they will be stored and used in the event detection process.

3.4.4 Event Detection

The main objective of sleep-related event detection is to achieve robust performance across different users, smartphone platforms and environments. iSleep adopts an adaptive event detection algorithm that adapts to the estimated noise model. First, acoustic features are extracted and normalized for each frame that is not detected as noise frame. Then, based on the normalized features ($\overline{rms}$, $\overline{rlh}$ and $\overline{var}$), frames are classified. Lastly, we apply operators in mathematical morphology [69] to the classification results to filter out false-positive and false-negative errors.

iSleep normalizes the features of each frame based on the current measurement of noise. Such a calibration process allows the system to adapt to different devices and environments. For example, the *rms* value of frame f is normalized as follows,

$$\overline{rms}(f) = \frac{rms(f) - mean(rms)}{std(rms)} \tag{3.7}$$

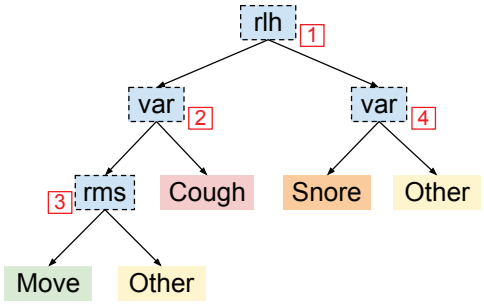


Figure 3.11 The decision tree for event detection.

where $\overline{rms}(f)$ is the normalized *rms*, $mean(rms)$ and $std(rms)$ are the current distribution features associated with *rms* extracted from the noise. Likewise, *rlh* and *var* are also normalized using the corresponding distribution features of the noise and the results are denoted as $\overline{rlh}(f)$ and $\overline{var}(f)$, respectively.

Fig. 3.10 shows the distribution of three normalized features. It is plotted from the data collected from 7 subjects using 3 different devices during a one-week experiment. The frames associated with each event are manually labeled and extracted. Then the three features of each frame are calculated and normalized using the current model of the noise. Four test subjects live close to the railway, and one subject lives in an apartment with automatic central A/C. In addition, the three devices have different noise profiles. However, Fig. 3.10 shows that, after normalization, the same events are grouped together and different groups are clearly separable.

Motivated by the results in Fig. 3.10, we design a decision-tree based classifier (shown in Fig. 3.11) to detect sleep-related events. Decision tree is robust to errors and widely used for classification problems with attribute-value pairs and discrete output values [55]. The splitting features and thresholds are determined based on the information gain calculated using entropy. Specifically, the entropy of a node T is given by

$$Entropy(T) = -\sum_{j} p(j) \cdot log(p(j)) \tag{3.8}$$

where $p(j)$ is the relative frequency of class j at node T. Since iSleep focuses on detecting three

sleep events, therefore, $j = 3$. After splitting node T into k nodes (T_1, T_2, ... T_k), the information gain is given by

$$G = Entropy(T) - [\sum_{j=1}^{k} \frac{n_i}{n} Entropy(T_j)] \tag{3.9}$$

where n_j is the number of samples in node T_j and n is the number of samples in node T. For each splitting, the system chooses the split that maximizes the information gain.

As the occurrence of most events results in clustered frames, isolated event frames are likely false positives. Therefore, after event frames are detected, we apply the opening operator in mathematical morphology [69] to filter out isolated frame events. Mathematical morphology is widely used in the identification of geometrical structures in image processing. Specifically, single or continuous event frames can be filtered out if the number of these continuous frames is less than the operator diameter (default value is 5). We apply the closing operator [69] to the resultant frame sequence after applying the opening operator, in order to connect those event areas with narrow gaps between them. This is because the narrow gap between two continuous event frame clusters is likely false negative. Specifically, if the length of the gap is less than the diameter of closing operator, the frames within the gap will be classified as event frames. Finally, we apply dilation operator [69] with the diameter of 2 frames to the continuous event frames. This will result in an expansion of 2 frames on both ends of the event frame sequences. The purpose of dilation is to ensure that the “edge" of this event is included.

Fig. 3.12 shows the event detection process in a typical scenario. The duration of the acoustic signal is 85 seconds, where the body movement of the user (18-26th second) is followed by a sequence of snoring events (38-83th second). Figure 3.12(a), (b) and (c) show the normalized features. We can observe that the increase of $\overline{rlh}$ clearly reflects the snoring event. The movement events usually have the similar $\overline{rlh}$ with noise, but higher $\overline{rms}$ and $\overline{var}$. In Fig. 3.12(d), the first plot shows the classification result for each frame. We can see that the movement and snoring events are detected correctly, but several noise frames are misclassified as event frames. The second plot in Fig. 3.12(d) shows the event detection result after we apply the opening, closing and dilation

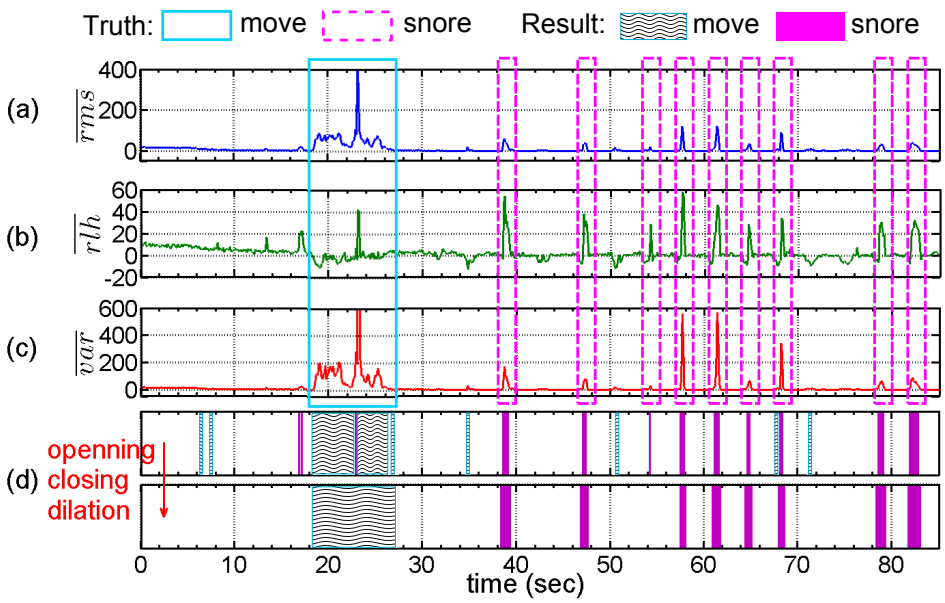


Figure 3.12 The event detection process of a 85-second acoustic signal that contains sound of body movement and snoring. (a), (b) and (c) show the features for each frame, respectively. (d) shows the event detection results before and after opening, closing and dilation operations.

operators. We can see that the isolated misclassified frames are removed, and the gap between two sequences of movement frames are closed. However, the snoring event at around 58 second is also filtered out. This is mainly because this particular snoring event is extremely short with very low intensity. As a result, only a single frame within this snoring event is detected as snoring frame. This frame is removed in the opening operation, causing a false negative error. Note that the detected snore events are used to calculate PSQI scores, which only require coarse-grain information about sleep-related events. Specifically, the detected snore will be used to automatically answer the question: 'During the past month, how often have you had trouble sleeping because you cough or snore loudly'. The options include 'not during the past month', 'less than once a week', 'once or twice a week' and 'three or more times a week'. As a result, a few misclassifications during a particular sleep will not affect the choice of answer. Therefore, misclassifying a weak snore or cough event is acceptable because its impact on the final sleep quality assessment is negligible.

3.4.5 Sleep Scoring

iSleep uses the detected sleep-related events to derive quantitative measures of sleep quality based on two criteria. One is actigraphy that only requires information about body movement of a whole-night sleep. The other is PSQI, where all the detected events are considered jointly for estimating the sleep quality.

In our implementation of actigraphy-based estimation, iSleep adopts similar method proposed in [75], where the sleep/wake state of a minute is determined by taking 4 previous minutes and 2 following minutes into account. The model takes the form:

$$D = P(W_{-4}A_{-4} + W_{-3}A_{-3} + W_{-2}A_{-2} + W_{-1}A_{-1} + W_0A_0 + W_{+1}A_{+1} + W_{+2}A_{+2}) \quad (3.10)$$

where P is a scale factor for the entire equation, W_{-i}, W_0 and W_{+i} represent the weighting factor for the previous minute, current minute and following minute, and A_{-i}, A_0 and A_{+i} indicate the activity scores for the previous minute, current minute and following minute, respectively. If $D \geq 1$, the state of the current is determined as wake, whereas $D \leq 1$ means the current minute is in sleep state. The model used in iSleep adopts the weighting factors suggested in [75]. It takes the following form:

$$D = 0.125(0.15A_{-4} + 0.15A_{-3} + 0.15A_{-2} + 0.08A_{-1} + 0.21A_0 + 0.12A_{+1} + 0.13A_{+2}) \quad (3.11)$$

where the activity score A is the number of frames associated with body movement in each minute. Fig. 3.13 shows the prediction of wake/sleep state over a 8-hour recording during sleep. The first plot shows the calculated activity scores for each minute. The second plot shows the calculated D value by using Eqn. 3.10. The last plot is the sleep/wake estimation result. We can see that the user changes from sleep state to wake state 4 times throughout the night. The duration of each wake state lasts around 10 minutes. The sleep efficiency is defined as the ratio of actual sleep time to total in-bed time:

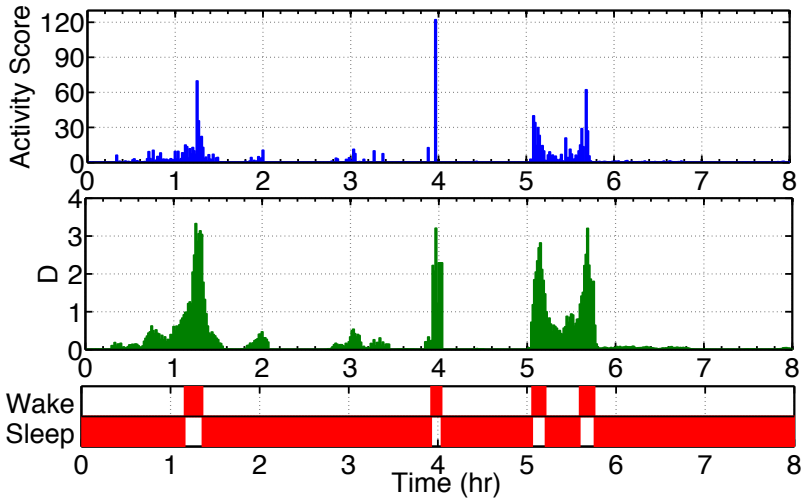


Figure 3.13 Actigraphy-based sleep/wake estimation based on recording of 8 hours.

Duration of Sleep	A4
Sleep Disturbance	B1, B2, B3, B4
Sleep Latency	A2, B1
Sleep Efficiency	A1, A3, A4

Table 3.1 The left column is the components in PSQI that can be derived from detected events. The right column is the metrics that are used to calculate the corresponding component score.

$$Sleep\ Efficiency = \frac{T_{sleep}}{T_{sleep} + T_{wake}} \tag{3.12}$$

For the long-term sleep quality estimation, iSleep calculates the scores of 4 components listed in Table 3.1 from PSQI. In order to calculate the component scores, iSleep measures the metrics listed in Table. 3.2 based on the detected events. However, some of the metrics used to calculate the score of *Sleep Disturbance* can not be measured by iSleep. For example, some of them are related to the bedroom's temperature, whether having dream, or feeling pain during sleep. As a result, instead of using the 9 metrics, iSleep uses only 4 metrics that can be inferred from the detected events and scales the score by multiplying $\frac{9}{4}$. The scoring rules of the other components are the same as specified in PSQI.

A1	Time to go bed at night
A2	Minutes taken to fall asleep
A3	Get-up time in the morning
A4	Hours of actual sleep per night
B1	Cannot sleep within 30 minites
B2	Wake up in the middle of the night or early morning
B3	Cannot breath comfortably
B4	Cough or snore loudly

Table 3.2 Metrics from PSQI that iSleep uses to estimate the sleep quality.

3.4.6 Monitoring the Sleep Quality of Two Users

In this section, we investigate the scenario where more than one users sleep in the same room. We assume each user runs a separate iSleep application on her/his own phone. Fig. 3.14 illustrates a typical scenario where two individuals sleep in the same bed and their phones are placed close to the bed, e.g., on two night stands. We assume that the phone is always closer to its owner than the other person. The major challenge for event detection in the two-user scenario is that the sound of one user can be captured by both phones, leading to inaccurate sleep quality assessment. Moreover, since the microphones of different phones have different sensitivities (as shown in Fig. 3.4), the sound intensity can not be used to reliably differentiate two users. Our key idea of differentiating the events of two users is to compare the $\overline{rms}$ associated with the events. Since $\overline{rms}$ indicates the loudness of the sound, it decreases when the phone is located further from the user. For instance, in Fig. 3.14, the distance between user A and phone B is longer than that between user A and phone A. Therefore, for the sound of user A, the $\overline{rms}$ calculated by phone A is larger than that by phone B.

In addition, since $\overline{rms}$ is normalized by the noise model measured by each phone, the impact of difference in microphones' sensitivities is mitigated. Fig. 3.15 shows the *rms* and $\overline{rms}$ of audio clip recorded by three devices (Nexus 4, Galaxy Nexus and Nexus 7) at the same distance. In Fig. 3.15(a), we can see that there exist substantial differences in *rms*, especially between Nexus 7 and the other two devices. However, Fig. 3.15(b) shows that, after normalizing *rms* by the current

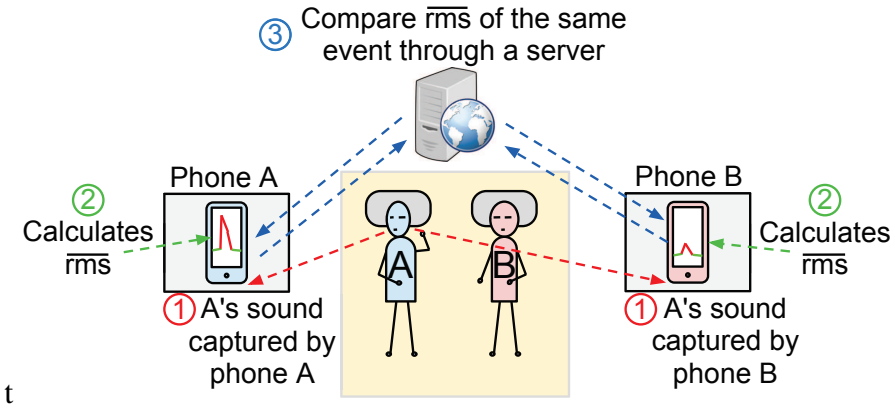


Figure 3.14 The process of differentiating the events of two users.

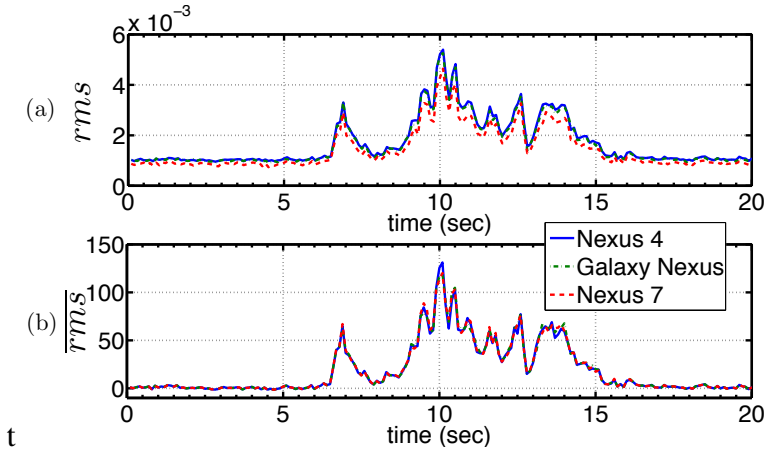


Figure 3.15 The *rms* and $\overline{rms}$ of an audio clip recorded by three different devices at the same distance. The audio clip contains several movement events from 6.5 to 16.5 sec.

noise model, the impact brought by different devices is minimized. In other words, two phones can compare their distances to the sound source according to $\overline{rms}$. Fig. 3.16 shows how the moving events of two users are detected, where the users and phones are located as shown in Fig. 3.14. We can see that, during user B's movement, the $\overline{rms}$ calculated by phone B is higher than that calculated by phone A. Thus iSleep is able to recognize two users by comparing the $\overline{rms}$ calculated by two phones.

Specifically, this process involves three steps. First, iSleep sends the computed $\overline{rms}$ and the timestamps of corresponding frames to a server. Second, the server groups the data sent by the phones in the same room. The phones in the same room generate very similar $\overline{rms}$ series (see

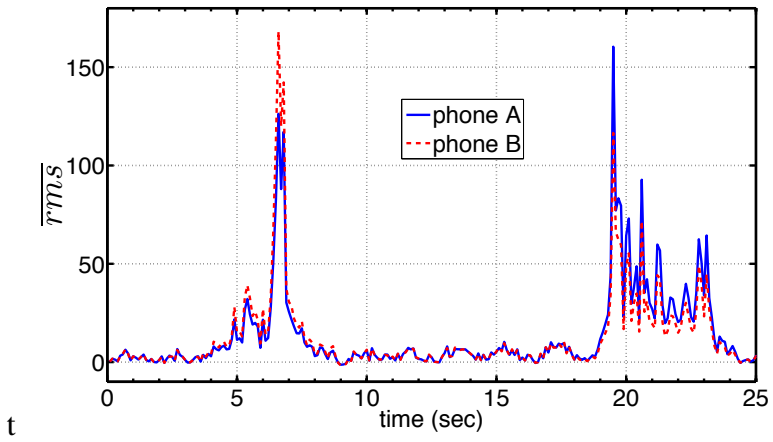


Figure 3.16 The $\overline{rms}$ of two users' movements captured by two phones in a real experiment. Users and phones are located as shown in Fig. 3.14. The distance between users is around 1.5 ft. The distance between the phone and the user is around 5 ft.

Fig. 3.15 and Fig. 3.16) although their amplitudes are different. This observation allows the server to easily identify the phones in the same room without knowing their exact locations. Other techniques, such as WiFi-based fingerprinting, may be combined with our method to improve the accuracy. Since the server cannot track the users' identities, there is no major privacy issue even if the server is able to tell which phones are in the same room. Third, for each event, the server compares the $\overline{rms}$ computed by two phones in the same room and sends back the larger value to each phone. We note that the amount of traffic between the phone and server is light. According to data collected from 7 iSleep users, the $\overline{rms}$ generated during a single night of sleep is about 1 kB.

3.5 Implementation

iSleep is implemented on Android 4.2.2 Jelly Bean. The application file has a size of around 1 MB and takes 2.7 MB storage on the phone after installed. It requires about 20 MB RAM allocation while running. The displaying and processing functions are implemented in separate threads to ensure the timeliness of acoustic sampling and processing.

iSleep samples the built-in microphone at 16 KHz. The samples are buffered and segmented into frames with the duration of 0.1 second. Based on the variance, iSleep detects non-noise frames

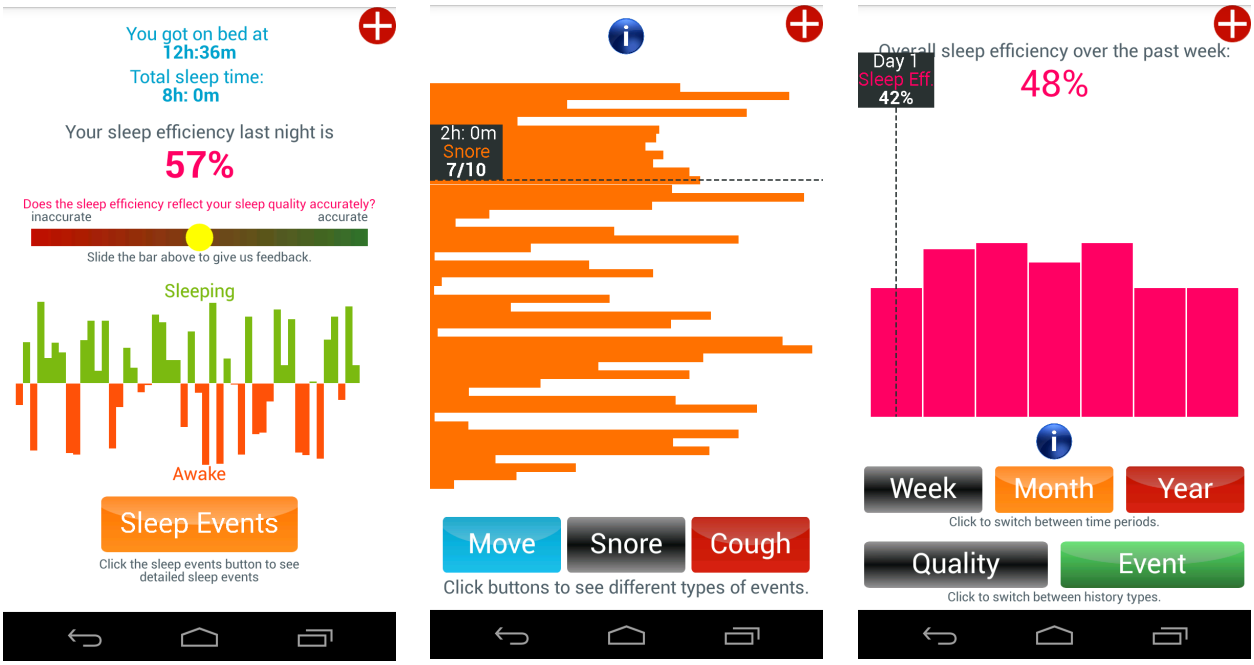


Figure 3.17 The user interface of iSleep. (a) The interface when iSleep is monitoring the user's sleep events. (b) The screen showing sleep efficiency and sleep states over night. (c) The screen showing the sleep events detected overnight and the normalized loudness of each event. (d) The screen showing the history of sleep efficiencies and events.

and noise-frames. Noise frames are used to estimate current noise distribution. Then, according to the noise distribution and features extracted from the non-noise frames, iSleep detects sleep-related events and saves them for further processing. Lastly, for each night, iSleep uses actigraphy to generate a short-term sleep quality report according to the movement events. For each week, iSleep estimates the long-term sleep quality according to PSQI and all the detected sleep events. To protect the users' privacy, iSleep only buffers the raw acoustic signal for the last 4 seconds and the buffered data is erased after the feature extraction.

We have released an initial version of iSleep on the Google Play Store [2]. The screen shots are shown in Fig. 3.17. The application is easy to use and understand. Before sleep, the user only needs to start the app and put the phone on the nightstand within 6 feet of the bed. iSleep prevents the CPU from sleeping, so that it can still keep running after the screen is turned off by pressing

the power button. After getting up, the user needs to stop the monitoring to see the sleep efficiency and detected sleep events. Within 6 days of release, iSleep has been installed by around 100 users from more than 9 countries on various Android devices. The feedbacks collected from the Google Play Store and the app show that users like to use iSleep to track their sleep quality and be aware of their sleep events.

3.6 Evaluation

In this section, we evaluate the performance of iSleep using experiments. Our primary results show that iSleep is able to effectively capture various sleep events and accurately estimate the sleep quality of the user.

3.6.1 Experimental Setting

We conduct three sets of experiments to evaluate the performance of iSleep. Section 3.6.2 presents the experimental results of a long-term experiment that involves 7 subjects and total 51 nights of sleep. All of the subjects are college/graduate student volunteers ranging from 20 to 30 years old, and 2 of the 7 subjects are male. Section 3.6.3 presents micro-benchmarks that evaluate the system performance under different settings and environmental factors.

Two metrics are used to quantify the performance of sleep event detection. *Event detection accuracy* (EDA) evaluates the accuracy of snoring and coughing detection. It is defined as the ratio of the number of correctly detected events to the total number of events. A snoring or coughing event is successfully detected as long as a subset of the frames associated with it is correctly classified. For instance, a snoring event containing 10 frames is considered as detected if iSleep correctly classifies at least one of these frames. This is mainly because iSleep only considers the number of occurrences of these two types of events when calculating PSQI scores. Moreover, the duration of a single snoring or coughing event is relatively short (typically around 1 second). Therefore, for our application, it is not necessary to detect the exact duration of these events.

sbj	nights	move(FDA)	snore(EDA)	cough(EDA)
1	13	91.7%	585/601(97.3%)	0/0
2	6	92.0%	0/0	0/0
3	12	89.8%	114/122(93.4%)	0/0
4	7	93.4%	0/0	0/0
5	4	94.1%	0/0	0/0
6	4	92.2%	0/0	85/85
7	5	90.1%	0/0	0/0
ttl	51	91.9%	699/723(96.7%)	85/85

Table 3.3 The event detection result based on the data collected from 7 subjects and total 51 nights of sleep.

Another performance metric used in our evaluation is *frame detection accuracy* (FDA). Different from EDA, FDA quantifies the performance of body movement detection. It is defined as the percentage of the correctly classified movement frames in all frames associated with the movement. This is because the calculation of activity score D (Eqn. 3.11) in actigraphy is based on the number of frames associated with body movement in each minute. Therefore, it is important to evaluate iSleep's accuracy in detecting the duration of body movement.

Our both evaluation metrics are based on true positive results, because the classification algorithm yields very few false negative and false positive results. The reason for this is two fold. First, the acoustic features we chose are very effective in differentiating different events. As a result, a sleep event is rarely mis-classified as another type of event (false negatives results). As shown in Table. 3.3, over 51 nights of sleep, around 6-10% of frames associated with movement are misclassified as noise, 48 out of 1446 (3.3%) snore events are misclassified as other events. Second, adaptive noise modeling and the mathematical morphology adopted by iSleep can effectively eliminate the noise frames that are mis-classified as sleep events (false positive results). Our long-term experiments show that, less than 20 minutes of various noises ($< 0.08\%$) are misclassified as sleep-related events, and no noise frames are misclassified as snore or cough events.

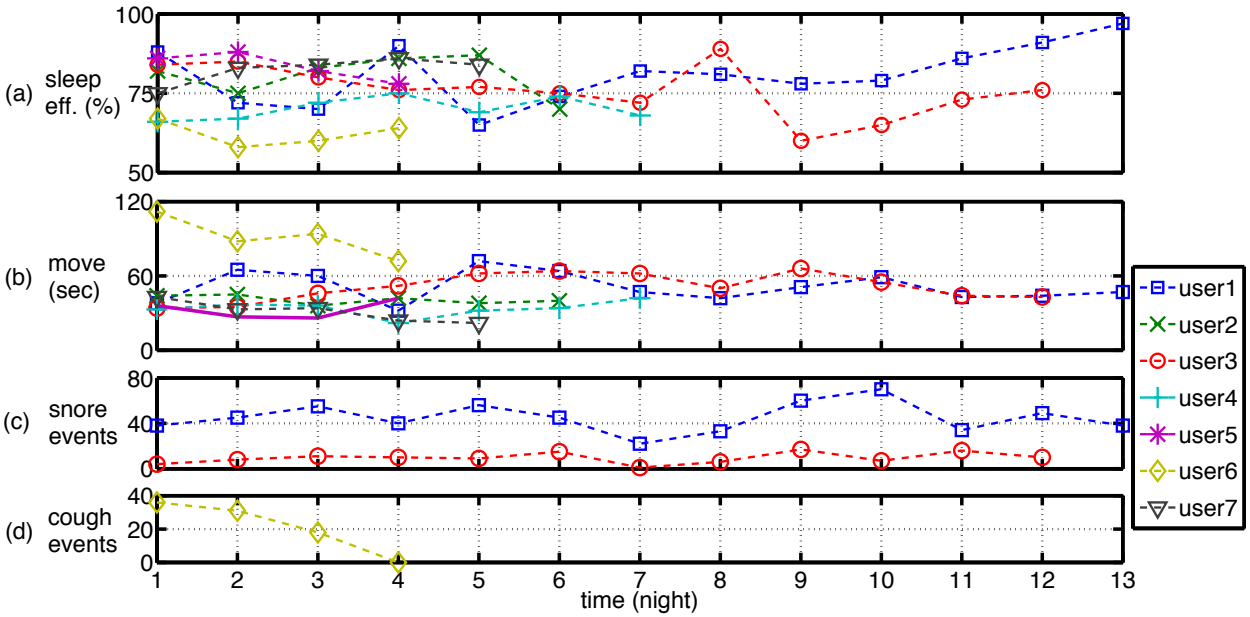


Figure 3.18 The sleep efficiency and sleep events of different users during the long-term experiment. (a) The sleep efficiency at each night. (b) The total time (in seconds) of body movement at each night. (c) The number of snoring events at each night. (d) The number of coughing events at each night.

3.6.2 Long-term Experiment

In this section, we present the result of a long-term experiment. 7 participants (2 males, 5 females) are recruited for data collection. The duration of the data collection for each participant varies from 3 to 14 days. There are totally 51 nights of sleep during the experiment.

The experimental platform used in data collection is composed of three components. First, two smartphones/tablets are put on both sides of the subject to collect acoustic data during sleep. The distance between the phone and the subject is around 5 feet, unless otherwise specified. The recorded audio clips are stored on the phones and retrieved for analysis after the experiment. Second, an omnidirectional microphone (Snowball USB Microphone) is attached on the headboard to collect the high-quality audio as ground truth of snoring and coughing events. A small laptop is connected with the microphone to store recorded audio clips. In order to minimize the impact of the noise from laptop fan, we place the laptop 16 feet away from the bed and connect it with the microphone using a long cable. Third, in order to record the ground truth of body movements, an

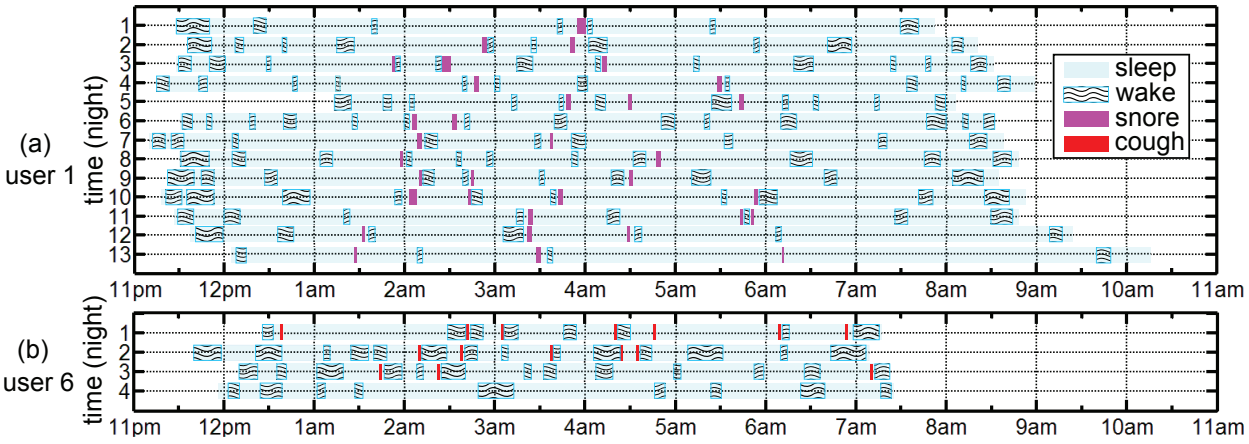


Figure 3.19 The sleep states and events of user 1 and 6 during the long-term experiment. The sleep states are calculated using the body movements based on actigraphy.

iPod touch is used to log the acceleration data of the user during sleep. In order to mitigate the impact of differences of mattresses, it is put inside a sport armband attached to the subject's lower leg.

After the data collection, we first synchronize the data captured by different devices. Then the snoring and coughing events are manually labeled from the high quality audio clips recorded by external microphones. The acceleration data collected by the iPod attached to the leg is used to obtain the ground truth for movement events. To evaluate the accuracy of sleep quality monitoring, each subject is asked to fill a PSQI questionnaire about how they feel about their sleeps during the period of data collection. The questionnaires are then used to correlate with the sleep quality measures calculated by iSleep.

Our evaluation is based on the data collected from 7 subjects and total 51-night sleeps. The overall detection results are shown in Table 3.3. We can see that the movement detection accuracies are relatively stable across different subjects, and the average FDA is 91.9%. The snoring detection accuracy is 97.3% and 93.4% for subject 1 and 3, respectively. The system achieves 100% coughing detection accuracy for subject 6.

Fig. 3.18 shows the sleep efficiency and sleep events detected by iSleep during the experiment. There are two snorers (user 1 and 3) out of seven subjects. Specifically, user 1 usually snores

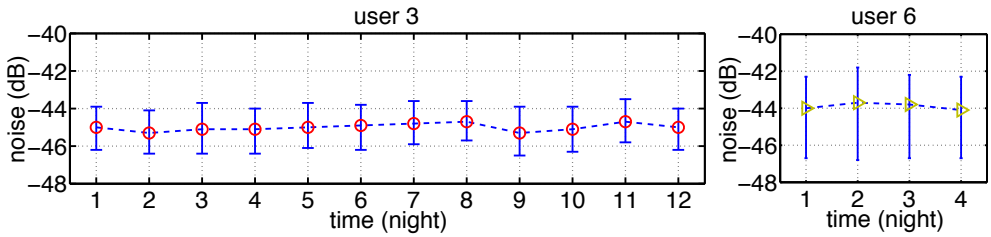


Figure 3.20 The noise levels (with 95% confidence interval) of two subjects during the long-term experiment.

periodically (every 2 or 3 seconds) for a duration of around one minute. User 3 snores more sporadically. Another observation is that coughing events are detected for user 6 during the first three days of experiment. This is due to the fact that user 6 happened to catch a cold. The number of coughing events gradually decreases every night as the user recovers. We can see that the users who snore or cough during sleep are more likely to have more dynamic and lower sleep efficiency. The main reason is that snores and coughs are usually followed by body movements, which indicate wakefulness of the user.

Fig. 3.19 shows the detailed sleep states and events detected by iSleep of user 1 and 6. We have several interesting observations: (1) Most of the snoring and coughing events are accompanied by body movements, which cause the subject to transition from sleep state to wake state. (2) User 1 usually snores during 2 to 5 am. (3) At the fifth night, user 1 got in bed about 1.5 hours later than she usually does. Due to the reduction of sleep time, her sleep efficiency is significantly lower than the other 12 nights. (4) The low sleep efficiency of user 6 is caused by her relatively short sleep time (around 7 hours) and frequent body movement due to the cold symptoms. The subjects are enthusiastic about these patterns of their sleep discovered in the experiment, and expressed interests in adopting tools like iSleep for long-term, daily sleep monitoring.

We observed different noise profiles in the sleeping environments of participants. Fig. 3.20 shows the noise intensity at different nights for User 3 and User 6. Although the noise level for each subject is relatively stable over time, the noise of User 6 is louder and substantially more dynamic than that of User 3. The high-quality audio data of external microphone confirms that

Subject	Duration of Sleep		Sleep Disturbance		Sleep Latency	
	PSQI	iSleep	PSQI	iSleep	PSQI	iSleep
1	1	1	1	1	1	1
3	2	2	1	1	1	1
4	1	1	1	1	0	**1***
5	1	1	1	1	1	1
6	3	3	3	**2***	1	1
7	2	**1***	1	1	0	0

Table 3.4 The comparison of PSQI scores provided by the subjects and computed by iSleep. The component score ranges from 0 (best) to 3 (worst). The scores of iSleep that do not match those from subjects' PSQI questionnaires are labeled by *.

this was due to the louder A/C in User 6's home. Despite the substantial differences in sleeping environments of different subjects, iSleep achieved consistently high detection accuracy, as shown in Table 3.3.

In order to evaluate the performance of measuring long-term sleep quality, we compare the 4 component scores (shown in Table 3.1) that are obtained from the subjects' PSQI questionnaires and iSleep. According to the scoring rules of PSQI, each component is scored on a scale of 0 to 3, where 0 means better and 3 means worse. iSleep calculates the component scores according to the same rules based on the detected sleep events. Table 3.4 shows the scores computed from subject questionnaires and by iSleep. We can observe that there is only one mismatch for Duration of Sleep, Sleep Disturbance, and Sleep Latency. And the score discrepancy for each mismatch is only one. This result demonstrates that iSleep can accurately predict users' answers to these questions based on objective assessment of sleep-related events. As a result, iSleep can be used as a reliable sleep diary that significantly reduces users' burden on remembering the details of their past sleeps.

3.6.3 Micro-benchmarks

This section presents a set of micro-benchmarks that evaluate the performance of iSleep under various distance and noise settings. We also evaluate iSleep in two-user scenarios and its processing

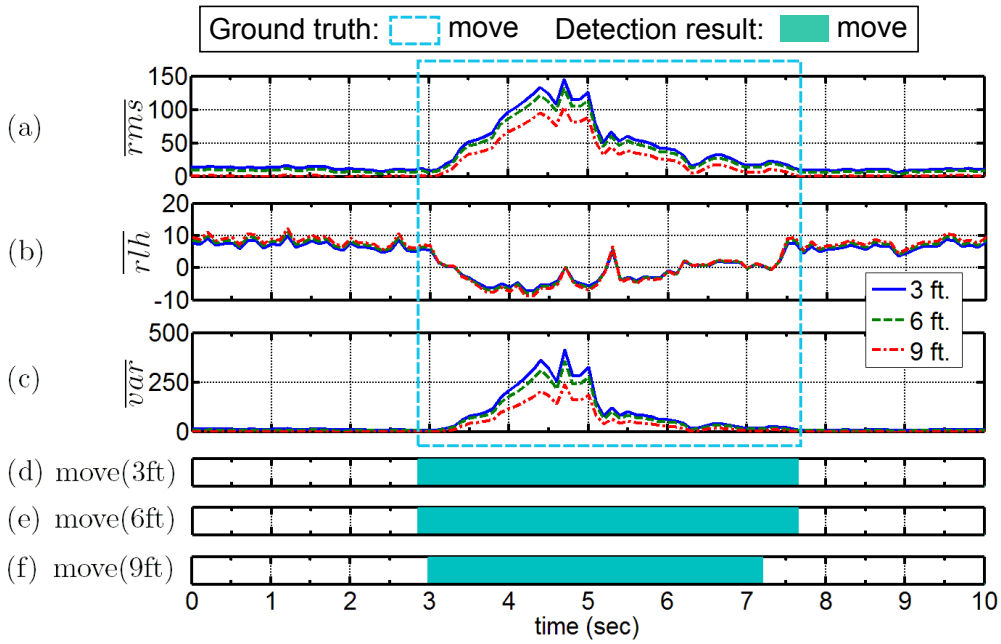


Figure 3.21 Event detection results based on a 10-second audio clip recorded at different distances (3, 6, and 9 feet). The movement events are labeled. (a), (b) and (c) show their features. (d), (e) and (f) are the detection results.

overhead and impact on battery lifetime.

In real scenarios, users likely place their phones at different distances from the bed. In order to evaluate iSleep's performance with respect to the distance between the phone and the user, we put phones at 3, 6 and 9 feet away from the user during sleep, respectively. The evaluation for each distance is based on a one-night experiment that lasts about 6.5 hours containing movement and snore events. The result of movement detection is shown in Fig. 3.22. We can see that increasing the distance between the phone and the user leads to lower movement detection accuracy. When the distances are 3 feet and 6 feet, the mis-classifications are mainly caused by minor leg or arm movements with relatively low sound intensity. However, when the distance is 9 feet, the sound intensity of movement events is substantially reduced. Another observation is that, the FDAs of different devices are relatively consistent at the same distance. This is because the acoustic features used in classification are normalized by the current noise model, making the detection robust against the differences in microphones' sensitivities.

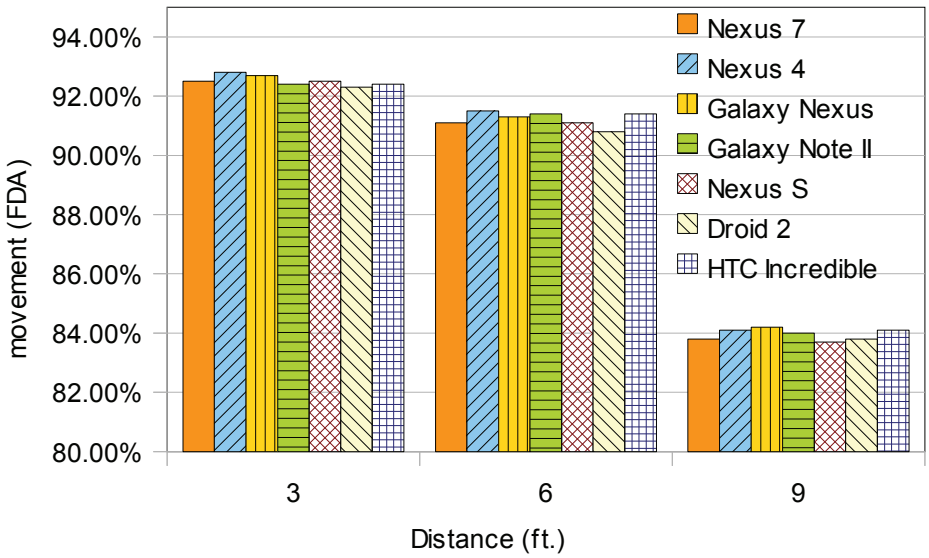


Figure 3.22 The impact of distance between the phone and the user on detection accuracy of movement. The acoustic data for each distance is collected from a 6.5-hour real experiment.

Fig. 3.21 shows the features and detection results of a 10-second audio clip captured from different distances. We can observe that the increase of distance leads to lower $\overline{rms}$ and $\overline{var}$. As a result, the frames on the edges of a movement event with low sound intensity are more likely to be mis-classified as noise. However, the detection of snore events is not affected by distance because of the significantly higher sound intensity.

Next, we investigate the accuracy of recognizing two users under different device and distance settings. iSleep compares the $\overline{rms}$ calculated by two devices to differentiate the events of different users. We focus on body movement events here because they have lower intensity than other events and hence are more difficult to differentiate. The recognition accuracy is defined as the percentage of movement frames which are correctly associated with the user. Six pairs of devices are used for each setting of distance difference. For each pair, one device is put 3 feet away from the user, while the other is located at 4, 4.5, 5, 5.5 and 6 feet away, respectively.

Fig. 3.23 shows the recognition accuracy based on $\overline{rms}$ of each frame. We can observe that, the accuracy raises with the difference of distances. Moreover, the pairs consisting of the same model of devices result in higher recognition accuracy (over 91%), because their microphones

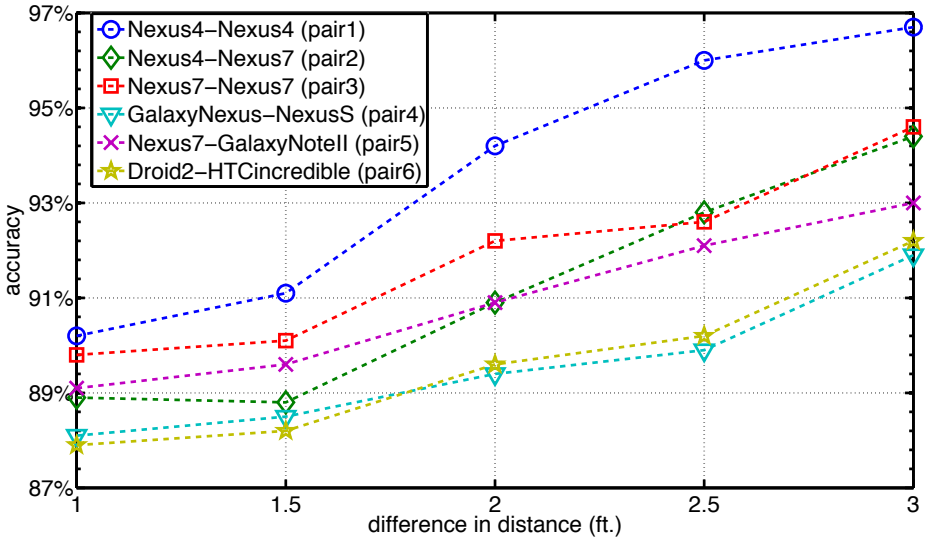


Figure 3.23 The accuracy of user recognition in two-user scenarios. In each experiment, six pairs of devices are used. The audio in each experiment contains movement events for a duration of around 10 minutes.

Device pair	1 ft.	1.5 ft.	2 ft.	2.5 ft.	3 ft.
pair 1	60/62	58/58	66/66	61/61	60/60
pair 2	59/62	57/58	65/66	61/61	60/60
pair 3	60/62	57/58	66/66	61/61	60/60
pair 4	57/62	55/58	65/66	60/61	60/60
pair 5	60/62	57/58	65/66	60/61	60/60
pair 6	57/62	56/58	64/66	60/61	59/60

Table 3.5 The user recognition accuracy by taking the majority vote for each movement event. The details devices are shown in Fig. 3.23.

have similar sensitivity. However, since each sleep-related event is composed of a sequence of frames, the recognition accuracy can be improved by taking a simple majority vote of the frames. As shown in Table 3.5, the average recognition accuracy is improved to 98%. The mis-recognitions mainly occur on the movement events with a short duration, such as a slight leg jerking for less than one second. When the distance difference is 1.5 feet or further, iSleep can achieve a recognition accuracy of more than 95%.

We now examine the impact of noise on the performance of iSleep. The evaluation is based on the real data containing body movements, snoring, coughing, and noises from various appliances

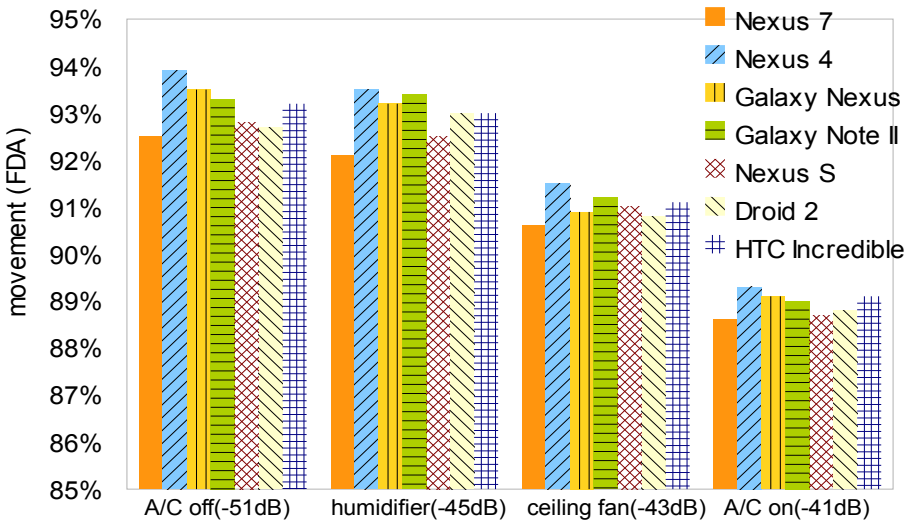


Figure 3.24 The impact of appliance noise on movement detection accuracy, based on the data from a real real experiment that lasts about 10 hours.

including a humidifier (around 9 feet away from the bed), a ceiling fan (around 8 feet above the bed) and the central A/C (two vents on the ceiling of the bedroom). The operational time of each of these appliances is at least 2.5 hours. iSleep can reliably detect all the snoring and coughing events under different noises. The result of movement detection is shown in Fig. 3.24. We can observe that the operation of appliances increases the average noise level, leading to an up to 10% drop in FDA. This is mainly because when the noise level rises, some movement frames with low sound intensity are mis-classified as noise. Specifically, iSleep can still achieve over 90% movement detection accuracy, while the ceiling fan and humidifier are operating.

Fig. 3.25 shows a typical event detection process in the presence of noise. The duration of the audio clip is 40 seconds, when the A/C starts operating at 0 second. We can observe that during the first 4 seconds, the $\overline{rlh}$ rises from around 0 to around 20, due to the low-frequency sound from A/C. Then the sound of the first 4 seconds is detected as noise, and used to update the noise model. As a result, the $\overline{rlh}$ falls back to around 0 at the 5th second. At the 9th second, the $\overline{rlh}$ rises again, due to the speed change of the A/C fan. iSleep detects sound from 10 to 14 seconds as noise, and updates the noise model at the 14th second.

We expect the smartphone to be connected to the charger when iSleep is used for the whole

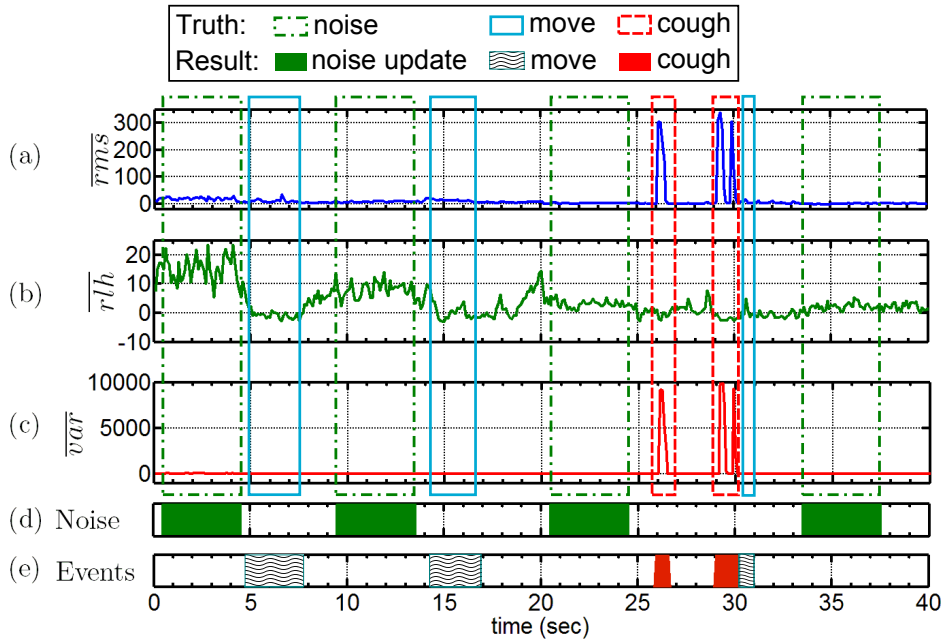


Figure 3.25 Event detection in the presence of operating A/C. (a), (b) and (c) show the acoustic features over time. (d) shows the detected noise frames that are used to update current noise model. (e) is the detection result.

night. However, users may forget to charge the phone, or would like to use iSleep during short naps without having to charging the phone. We now evaluate the processing time of each component of iSleep and the system energy consumption. The evaluation results based on data collected from 5 devices overnight are shown in Table 3.6. We can see that the feature extraction component consumes the most processing time among all components, since three features need to be computed for each frame. Thanks to the light-weight decision-tree based classifier, the event detection component only consumes around 0.2% of total CPU time. We also evaluate energy consumption of iSleep based on the battery usage data from the Android system. Since the screen is turned off, computation and microphone sampling are the major sources of power consumption. On average, iSleep consumes around 4% battery per hour (excluding the system consumption of Android). This result suggests that, a fully charged phone running iSleep likely survives a full night of usage without connecting to the charger.

Phones	ND	FE	ED	Total	% of CPU
Nexus 7	30ms	38ms	0.15ms	68.15ms	1.7%
Nexus 4	28ms	36ms	0.13ms	64.13ms	1.6%
Nexus S	67ms	88ms	0.27ms	155.27ms	3.9%
G. Nexus	40ms	53ms	0.15ms	93.15ms	2.3%
G. Note II	33ms	40ms	0.15ms	73.15ms	1.8%

Table 3.6 The average CPU time consumed by different components of iSleep to process 4-second acoustic data. (ND: noise detection, FE: feature extraction, ED: event detection).

This section presents a preliminary evaluation based on the data of real iSleep users. We collected data from the Android phones that downloaded and installed iSleep from Google Play Store during the first week after the release of iSleep. Although there were more than 100 installs, as expected, many users opted out the data collection. The information collected include users' ratings on the accuracy of sleep efficiency computed by iSleep as well as the numbers of various events detected during each night (no raw acoustic data was collected). On the screen of monitoring results, iSleep shows a slide bar (see Fig. 3.17) that allows the user to rate the accuracy of the sleep efficiency measured by iSleep on a scale of 0 (not accurate) to 100 (very accurate). The average of 25 scores on sleep efficiency from users is above 85%. Fig. 3.26 shows the results of four users randomly chosen from those who participated in the data collection. We can see that, both the total in-bed time and actual sleep time are relatively consistent for the same user, reflecting the user's normal sleep behavior. A detailed analysis of the results also suggests that the shorter sleep time is usually caused by either snoring or extensive body movement. Another observation by correlating the sleep efficiency and user ratings is that, users are more likely to give low feedback scores when the measured sleep efficiency is low.

3.7 Summary

We have described the design, implementation, and evaluation of iSleep – a practical system to monitor an individual's sleep quality using off-the-shelf smartphone. Compared with existing solutions, iSleep is very easy to use and unobtrusive. iSleep uses the built-in microphone of the

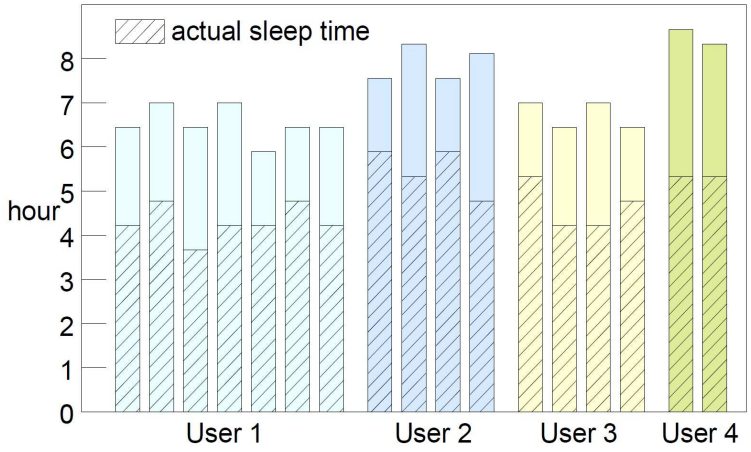


Figure 3.26 The actual sleep times and in-bed times of 4 iSleep App users.

smartphone to detect the events that are closely related to sleep quality, including body movement, couch and snore, and infers quantitative measures of sleep quality based on actigraphy and Pittsburgh Sleep Quality Index (PSQI). We have evaluated iSleep extensively in a long-term experiment that involves 7 participants and total 51 nights of sleep. Our results show that iSleep achieves above 90% accuracy for sleep-rated event classification in a different settings. The fine-grained sleep profile measured by iSleep also enabled users to track details of sleep events over time and discover irregular sleep patterns.

The high-rate microphone sampling is a major source of energy consumption. We will investigate an adaptive sampling scheme in which the microphone is sampled at a low rate, and only sampled at a higher rate when a potential event is detected. Environmental factors such as room temperature play an important role in quality of sleep. We plan to integrate iSleep with tools that can monitor sleep environments [41]. This will enable in-depth analysis of causes of interrupted sleep and irregular sleep patterns, providing important information for healthcare providers to find trends related to certain diseases.

3.8 Disclaimer

We have obtained IRB approval (IRB# 12-1178) from the Institutional Review Board at Michigan State University for the use of human subjects in this study, including the controlled experiments and the data collection from our mobile app.

CHAPTER 4

LEVERAGING PHYSIOLOGICAL MODEL FOR RUNNING RHYTHM MONITORING

One of the key challenges of smartphone-based physiological sensing is its accuracy. As we learned in the process of designing and evaluating iSleep, various environmental noises are able to heavily impact the performance of the monitoring system by causing false alarms. This is especially true for acoustic-based sensing systems like iSleep. On one hand, compared with other approaches like motion-based sensing, leveraging acoustic signal makes the system less obtrusive by freeing users from constantly carrying the smartphone with them. On the other hand, the sampled data is susceptible to significantly more environmental noises.

In this chapter, we propose a novel idea that helps improve the mobile sensing accuracy by leveraging related physiological model. In the following, we demonstrate the integration of mobile sensing and physiological model by presenting the design of a smartphone-based exercise monitoring system. Compared with existing exercise monitoring systems, RunBuddy differentiates itself as the first smartphone-based system that aims to continually monitor user's running rhythms. It is designed to be a convenient and low-lost monitor. RunBuddy only utilizes commodity devices including smartphone and bluetooth headset, without relying on any custom hardware. In addition, RunBuddy is also easy-to-use and truly unobtrusive. Users only need to wear the bluetooth headset and carry the smartphone during running.

4.1 Introduction

In animals and humans alike, respiration is often coupled to locomotion in order to efficiently sustain endurance exercise [20]. When moving about, the locomotory system is responsible for meeting the mechanical requirements. It needs to generate forward motion while maintaining upright stability. The respiratory system supplies necessary amount of oxygen for metabolism, and removes metabolic byproducts from the circulatory system. These two systems are both critical

to locomotion, and do not work independently of one another. In the case of running, the strides can be viewed as a driver signal, and the breaths are the dependent signal that varies based on the frequency of the driver signal.

The running rhythm, which characterizes the coordination between breathing and strides, varies throughout the run depending on factors such as the duration and intensity of the exercise, and training and fitness level of the runner [15]. The rhythm can also be improved over time by training: a more stable rhythm has been found in both experienced runners [15, 54] and cyclists [70, 42]. Interestingly, sound-induced stablization of such rhythm – listening to an external auditory stimulus with a proper tempo during rhythmic exercise – has been shown to result in significant reduction in oxygen consumption and improvement in exercise efficiency [34].

Unfortunately, to date, there has been no convenient and unobtrusive way of measuring running rhythm continuously. Cardiopulmonary exercise testing (CPET) is a widely adopted clinical tool to evaluate exercise capacity. It provides an analysis of respiratory gas exchange and cardiac function during exercise. However, CPET is usually limited to hospitals and clinics, due to its complicated procedure and high cost (about $\$20,000$ /unit). Recently, several wireless CPET products have been developed. For example, Oxycon Mobile [62] integrates various lightweight sensors into a vest, which is worn by the subject during the exercise. However, designed for short-term evaluation of professional training, such devices are too bulky to wear for everyday use.

This chapter presents RunBuddy – the first smartphone-based system for running rhythm monitoring. RunBuddy is designed to be a convenient and unobtrusive exercise feedback system, which provides the user continuous measurement of their running rhythm. It only utilizes commodity devices including smartphone and bluetooth headset which is often worn by runners to listen to music and make phone calls, without relying on any custom hardware. Moreover, RunBuddy provides fine-grained and continuous measurement of the user's running rhythm, which can improve the exercise experience and help the user better understand his/her exercise self-efficacy.

Specifically, RunBuddy measures the user's running rhythm using a physiological metric called

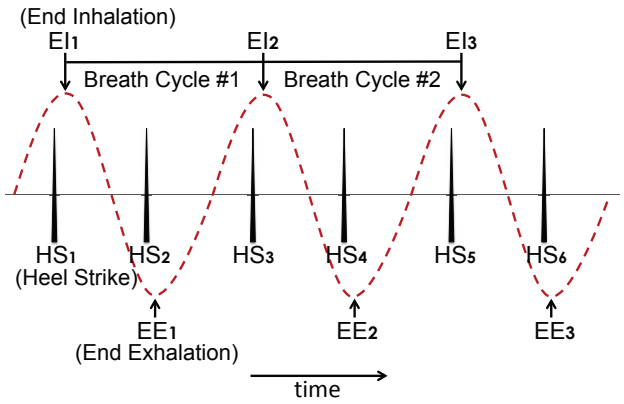


Figure 4.1 An example of LRC with a 2:1 stride to breath ratio.

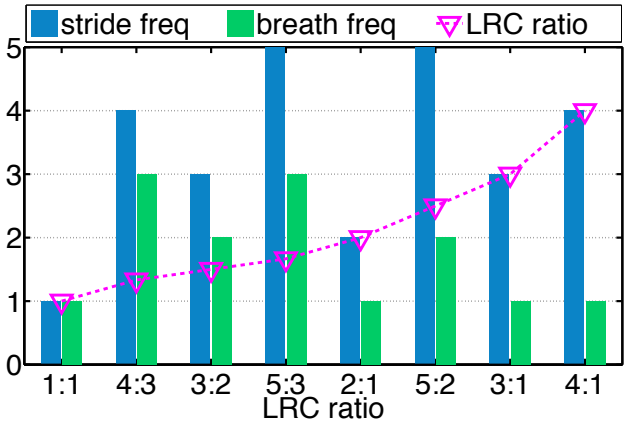


Figure 4.2 LRC ratios that have been observed in humans while running [72].

Locomotor Respiratory Coupling (LRC) [20] [1]. Figure 4.1 illustrates the LRC in humans with a stride to breath ratio of 2:1. The breath and stride signals are simulated using sinusoid wave and pulse signal, respectively. The LRC ratio can be calculated as the number of of strides within each breath cycle. LRC has also been found in a number of rhythmic exercises including cycling [64], rowing [52], running [20] and walking [48]. In general, the degree of coordination is higher during running than during other activities (e.g., walking and cycling) [16]. Specifically, LRC is defined as frequency and phase locking between two periodic systems (i.e., locomotory and respiratory system). Horses, for example, have an almost fixed stride to breath ratio of 1:1 [20]. For humans, a small number of low integer coupling ratios are observed [20], which are shown in Figure 4.2.

[1] We use the terms “running rhythm" and “LRC ratio" interchangeably hereafter.

In order to be convenient and unobtrusive, RunBuddy analyzes acoustic samples from the bluetooth headset worn by the user to detect breathing. However, the sound of breathing typically has very low intensity and is susceptible to various interferences from the environment (e.g., music in the gym, sound of wind or traffic, and etc.). Moreover, the breathing pattern can vary significantly among different individuals during running, in terms of the loudness, acoustic frequency range. Lastly, in order to preserve users' privacy and provide real-time feedback, the system must process the data on the fly, and should not store or transmit raw sound samples. Therefore, the breath detection algorithms must be lightweight while maintaining satisfactory measurement accuracy.

To address above challenges, we propose a novel approach to the design of RunBuddy, which integrates ambient sensing based on accelerometer and microphone, and a physiological model LRC to improve the accuracy of running rhythm monitoring. LRC is a quantitative measure of the running rhythm: the coordination of locomotory and respiratory systems that are both critical to locomotion. According to the LRC theory, there exists a small number of low-integer coupling ratios (e.g., 2:1) between the stride and breathing frequencies [20]. RunBuddy leverages the LRC theory to calibrate the running rhythm measurement. Specifically, we develop a lightweight signal processing pipeline to detect breathing from the acoustic samples of bluetooth microphone. Due to the low sound intensity of breathing and the impact of environment noises, the acoustic breath detection results may be highly inaccurate. To address this issue, we detect strides through smartphone accelerometer, and correlate strides, possible LRC ratios, and preliminary acoustic breath results together to calibrate the running rhythm measurement. The LRC has been found in a number of rhythmic exercises including cycling [64], rowing [52], running [20] and walking [48]. Therefore, the approach of RunBuddy can potentially be applied to monitor the rhythm of various rhythmic exercises.

We believe a personal, obstructive running rhythm monitoring system like RunBuddy holds great potentials to improve users' exercise experience and self-efficacy. Many beginning runners and people who are usually sedentary lose interest or even get frustrated quickly because they can not catch their breath during running, due to exercise hyperpnea [10]. By monitoring their running

Figure 4.3 A typical setting of RunBuddy. The user is required to wear a bluetooth headset and carry a smartphone while running.

rhythm in real-time, the system could potentially guide them to achieve a more comfortable rhythm by playing music with proper tempos (e.g., by adjusting the workout soundtrack) [12, 40, 17]. Moreover, based on the measured running rhythm, the system can provide an analysis of users' physiological profile, such as the stability of LRC ratios during running which is a good indicator of fitness level [54].

4.2 System Requirements and Challenges

RunBuddy is designed to be a convenient and unobtrusive exercise feedback system, which provides the user continuous monitoring of their running rhythm. Specifically, RunBuddy measures the running rhythm in terms of LRC ratio in a time window (e.g., 10 seconds) continuously during running. Based on the measured LRC and stride frequency, RunBuddy can also estimate real-time breathing frequency (number of breaths per minute). However, we believe LRC is a more desirable metric for runners since it quantifies the coordination between breathing and stride. Research has shown that LRC is a good indicator of fitness level [53].

RunBuddy is designed to meet the following requirements: (1) Since RunBuddy operates during running, it needs to be unobtrusive. The user should not feel any kind of uncomfort or added burden when using the system. (2) RunBuddy needs to provide fine-grained and continuous measurement of the user's running rhythm, which can improve the exercise experience and help the user better understand their exercise. (3) Such measurement needs to be robust and accurate across different users and exercise environments. (4) The system needs to strictly protect the users' privacy. Since the system relies on sound recognition in breath detection, users might have concerns about possible privacy breach, such as the content of their conversation during running being recorded. Therefore, instead of sending the raw sensor data to a remote server, the system should process the data on the fly and only keep the data that is related to the user's exercise (e.g., breathing and stride detection result over time). Moreover, online processing is also required by applications that need to provide real-time feedback (e.g., music suggestions) to the user.

To meet these requirements, three major challenges need to be addressed in developing RunBuddy. First, in order to make RunBuddy convenient and unobtrusive, the system samples and analyzes acoustic signals from the bluetooth headset worn by the user to detect breath. However, the sound of breathing has low intensity, making it challenging to capture. Second, the sound of breathing can be different among different individuals, in terms of its loudness and frequency range. Moreover, noises from the exercise environment such as music in the gym may also have a major impact on the accuracy of breath detection. Lastly, RunBuddy must adopt extremely lightweight signal processing algorithms as it needs to process high-rate sensor stream (e.g., 16 KHz sound and 200 Hz acceleration) in real-time.

4.3 System Overview and Applications

RunBuddy aims to continuously monitor use's running rhythm. It requires no extra hardware except a bluetooth headset and the smartphone. Figure 4.4 shows the overview of RunBuddy. The system consists of four major components, namely, *breath detection*, *training*, *stride detection*, and *LRC-based correlation*.

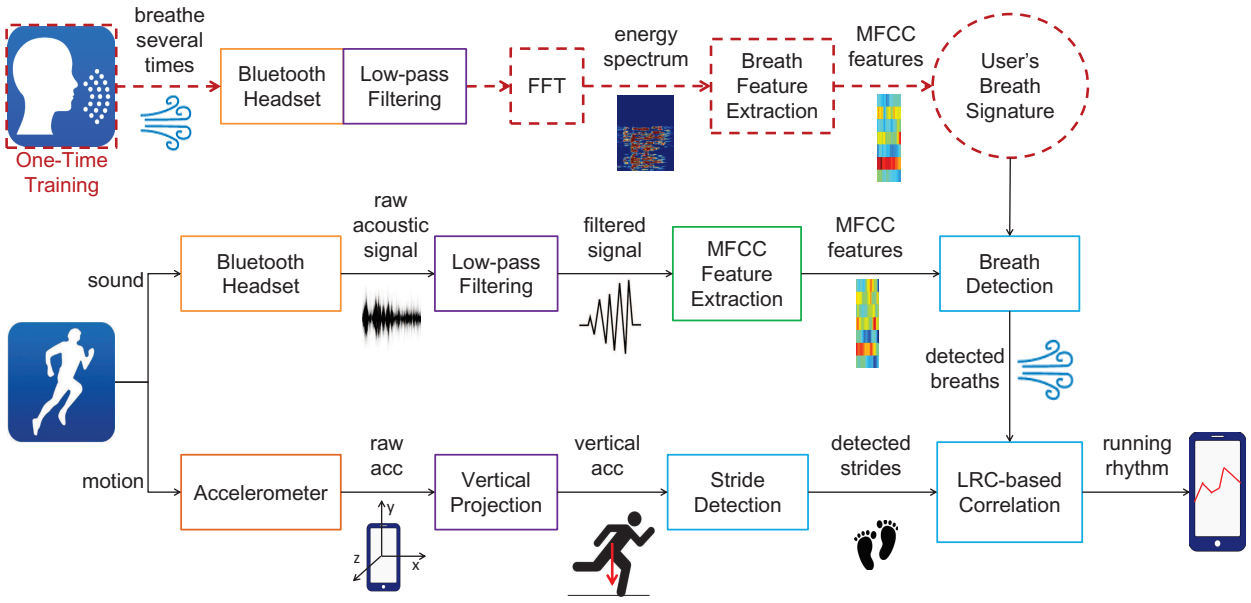


Figure 4.4 Overview of the RunBuddy system. RunBuddy requires a one-time training, and comprises 3 major components: breath detection, stride detection and LRC-based calibration.

The key novelty of RunBuddy lies in the integration of ambient sensing and physiological models to improve the accuracy of running rhythm detection. Specifically, RunBuddy detects breaths based on its unique sound features. Due to the low sound intensity of breath and the impact of environment noises (e.g., music in gym and wind outdoors), the acoustic breath detection result may be highly inaccurate, leading to poor accuracy in LRC ratio measurement. Leveraging the LRC theory, RunBuddy addresses this challenge by correlating the detected breath with the more reliable stride detection result to find the most likely LRC ratio.

The breath detection is based on sound, which is continuously collected through the bluetooth headset worn by the user during running. First, the raw acoustic signal is processed by a low-pass filter, where high-frequency noises are filtered out. Second, RunBuddy frames the filtered signal and extracts MFCC features from each frame. It then uses the first 7 features (12 variables in total) for *breath detection*. In order to improve the accuracy of breath detection, RunBuddy requires a straightforward one-time training to capture the sound features of the breath from a particular user.

To complete the training, the user only needs to breathe several times. The training sound captured by the bluetooth headset is fed into the same low-pass filter used for breath detection. Next, the energy spectrum of the filtered signal is calculated using *FFT*. Then, the energy spectrum over time is fed to *breath feature extraction*, where RunBuddy detects frames that contain the user's breath based the energy, and extracts their MFCC features. These extracted features are considered as the *breath signature*, which represents the unique sound features of the user's breath. At runtime, RunBuddy is able to detect a possible breath by calculating the similarity between extracted MFCC features and the user's breath signature. However, the breath detection result at this stage may be inaccurate due to the impact of environmental noise. As a key novelty, RunBuddy calibrates such preliminary breath detection results by correlating with strides based on possible LRC ratios.

To detect strides, RunBuddy first captures motion through sampling the built-in accelerometer. The collected acceleration is first processed to obtain the vertical acceleration (with the same direction of gravity), by projecting the raw data onto the global coordination system. Then the vertical acceleration is taken as the input of *stride detection*, where several filters are employed to enhance the performance of the detection [2]. In the *LRC-based correlation*, RunBuddy utilizes the detected strides to calibrate the breath detection results, according to LRC. Based on the theory of LRC, there exists a small number of low-integer coupling ratios (e.g., 2:1 and 3:2) between the stride frequency and the breathing frequency. The coupling is especially strong during running, due to the more frequent leg movement, and the increased intensity. RunBuddy is able to reliably measure the running rhythm, by correlating the detected strides and breathing based on possible LRC ratios for humans.

As running rhythm is essential in reflecting the user's physical state, RunBuddy has great potentials to benefit users' exercise experience in many scenarios. First, as suggested by previous study [54], the ability of maintaining a stable running rhythm can potentially be used to infer the runner's fitness level. For example, compared to non-runners, regular runners are able to maintain a more stable LRC ratio during running [15, 54]. Second, based on the running rhythm, the

[2]Some exercise monitoring applications may already include stride detection algorithms. In such a case, RunBuddy may directly use the real-time output of these algorithms.

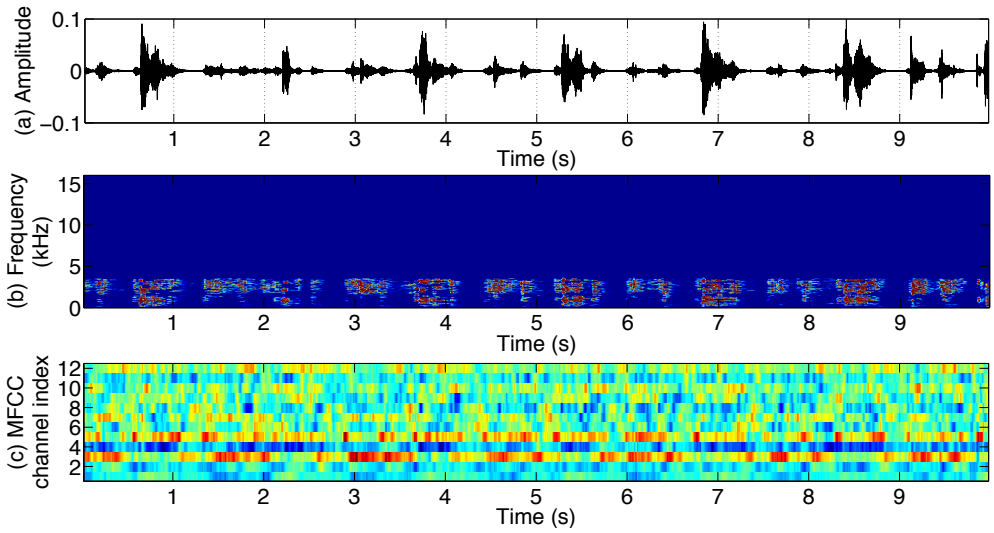


Figure 4.5 An example of MFCC extraction from a 10-second sound clip recorded while running. (a) The waveform of the sound clip. (b) The energy spectrum of the sound over time. (c) The extracted MFCC features over time.

system could potentially provide real-time and personalized feedbacks such as music suggestions to enhance users' exercise experience. For instance, many new runners lose interest or even get frustrated because they can not catch their breath during running. By monitoring their running rhythm in real-time, the system could help them achieve a more comfortable rhythm by playing music with proper tempos (e.g., by adjusting the workout soundtrack) [12, 40, 17]. Once feeling comfortable, new runners will be more willing to engage in running.

4.4 System Design

In this section, we first describe the process of extracting acoustic features used for breath detection. Next, we explain the training process and how we utilize training data to detect breath. Then, our approach to recognizing stride from acceleration is described. Lastly, we present the LRC-based correlation, where we leverage the correlation of detected strides and breathing to improve the accuracy of running rhythm measurement.

4.4.1 Acoustic Feature Extraction

During running, the system continuously collects acoustic signal through the microphone at a sampling rate of 16 *kHz*. Prior to feature extraction, we use a low-pass Butterworth filter of order 2 to suppress high frequency noise in the raw data. Since the typical sound frequency of human breath falls in the range of 500 to $3,500$ *Hz*, we set the cutoff frequency of the filter at $3,500$ *Hz*. After pre-processing, the acoustic signal is framed using a moving window. Each *frame* includes 40 *ms* of acoustic signal and has 30 *ms* overlap with nearby frames. For each frame, we calculate the features of mel-frequency cepstral coefficients (MFCCs) within the frequency range of 100 to $5,000$ *Hz*. The energies of the first 7 MFCC channels are used as the feature vector for breath detection.

Figure 4.5 shows an example of MFCC feature extraction based on a 10-second acoustic data containing several instances of breath cycles (each cycle consisting of one nose inhalation and one mouth exhalation). From the energy spectrum (Figure 4.5(b)), we can see that energy is mostly distributed below $3,500$ *Hz*. Figure 4.5(c) shows the extracted MFCC features over time. We can observe that breath is associated with certain energy patterns in lower MFCC channels.

4.4.2 Training for Breath Detection

A one-time training process is necessary for sound-based breath detection. This is primarily because the sound of breathing is different among individuals, in terms of its duration, strength and frequency range, resulting in diverse acoustic profiles. Therefore, it is necessary to customize the breath detection for each user based on training. The training process is designed to be simple and easy to follow. Before the first time use of the system, the user is required to breath several times in a relatively quiet environment. This ensures that the system can capture the sound of breathing with as little noise as possible. Moreover, the system can automatically detect the sound of breathing. This spares the user's effort of manually labeling the breath events from raw data. The detected breath events are then used to generate the user's breath signature represented by MFCC features.

4.4.3 Breath Detection

The goal of breath detection is to recognize acoustic frames that contain breath. Basically, it estimates the likelihood of a frame containing breath, by calculating the similarity between its feature vector and the training data. Specifically, let $bv_1, bv_2 ... bv_n$ be the feature vectors of n breath frames extracted from the training data. First, we calculate the mean vector $\overline{bv}$ of all the breath frames as follows,

$$\overline{bv} = \frac{\sum_{i=1}^{n} bv_i}{n} \tag{4.1}$$

Then, we calculate the cosine similarity $S(bv_i, \overline{bv})$ between bv_i and $\overline{bv}$. The calculation is given below.

$$S(bv_i, \overline{bv}) = \frac{bv_i \cdot \overline{bv}}{\| bv_i \| \| \overline{bv} \|} \tag{4.2}$$

The resulting cosine similarity ranges from -1 (the minimum similarity) to 1 (the maximum similarity). The cosine similarity has been widely used in information retrieval to measure the similarity between two texts. We use cosine similarity to measure the similarity between two vectors. Since cosine similarity is independent of magnitudes of acoustic signals, it is effective at capturing the characteristic of MFCC features and robust against environmental noises. Lastly, we use the minimum value of the similarities as the threshold T. Given $\overline{bv}$ and threshold T from the training data, we define the likelihood that a frame with feature vector bv^*, which is captured at runtime, contains breath as,

$$L(bv^*, \overline{bv}, T) = \begin{cases} \frac{S(bv^*, \overline{bv}) - T}{1 - T} & \text{if } S(bv^*, \overline{bv}) > 0 \\ 0 & \text{if } S(bv^*, \overline{bv}) \leq 0 \end{cases}$$

where $S(bv^*, \overline{bv})$ is the cosine similarity between bv^* and $\overline{bv}$. Figure 4.6 shows an example of breath detection as well as the ground truth based on a 20-second acoustic data. The subject alternately uses nose and mouth to breathe. We can see that even though there exist some detection errors, this method is able to provide reasonable estimation of breath event.

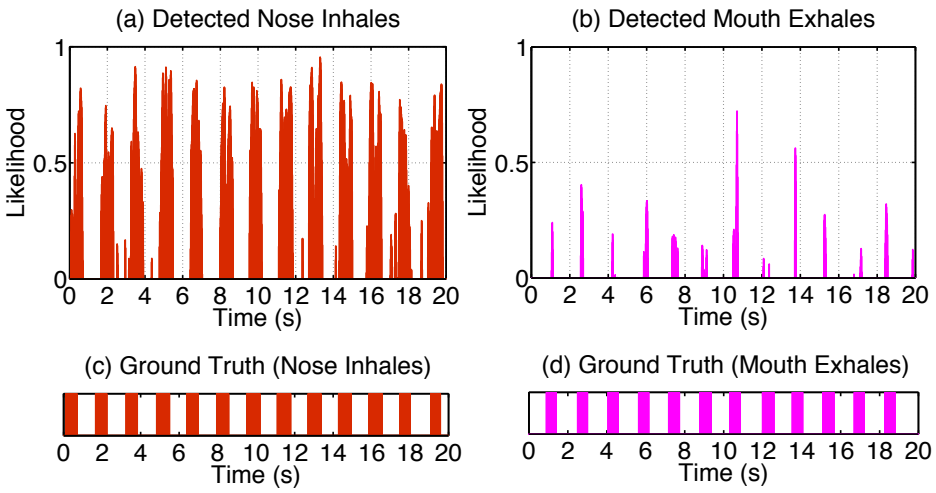


Figure 4.6 (a) and (b) show the likelihood of the detected breathes. (c) and (d) show the ground truth of the breathes.

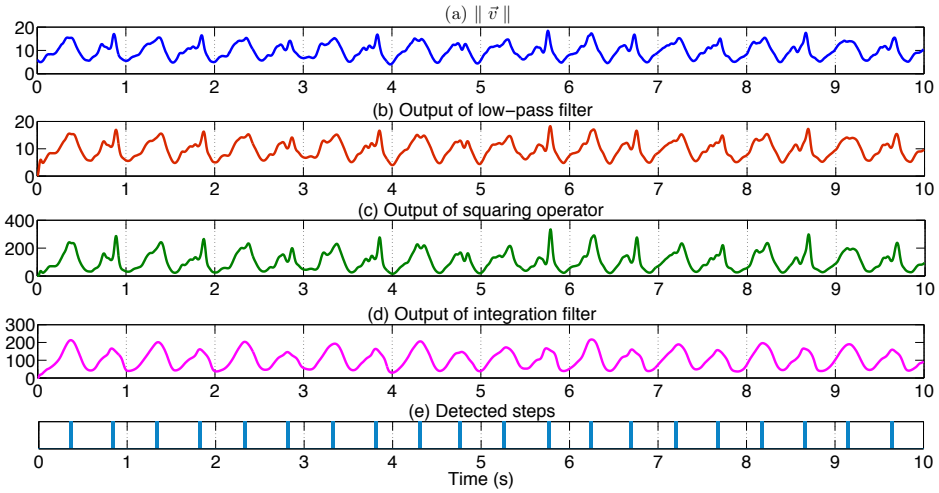


Figure 4.7 An example of stride detection.

4.4.4 Stride Detection

Stride detection aims to recognize strides from collected acceleration data. The stride detection method adopted by RunBuddy is based on searching for peaks in the change of acceleration during running. Although this method is similar to several existing methods [51, 19], we include the details here for completeness. Specifically, RunBuddy first projects the sampled acceleration data onto global coordination system. Then it detects strides by searching for peaks in the vertical acceleration (the gravity direction).

By projecting acceleration onto a global coordinate system, we don't have to consider the phone's orientation change in the following processing. Moreover, the vertical acceleration is more sensitive to the shock caused by each heel strike. Prior to projection, RunBuddy estimates the direction of the gravity by averaging the acceleration readings along each axis for a short period of time (e.g., 5 seconds) [56]. After estimating the gravity direction, we project collected acceleration data on to global coordinate system. Let $\vec{a}$ be the acceleration reading and $\vec{g}$ the estimated gravity. The vector representing vertical acceleration $\vec{v}$ can be calculated as follows,

$$\vec{v} = \vec{a} \cdot \frac{\vec{g}}{\| \vec{g} \|} \tag{4.3}$$

We can also get the horizontal acceleration $\vec{h} = \vec{a} - \vec{v}$. Next, we feed $\| \vec{v} \|$ into a series of filters to detect stride which contains one left heel strike and one right heel strike. Since the phone is attached to the user during running, our stride detection method is based the assumption that each major peak in the $\| \vec{v} \|$ is caused by one heel strike. Therefore, detecting stride in this scenario is essentially a peak detection problem. In order to improve the accuracy, prior to the peak detection, we process $\| \vec{v} \|$ with several filters included in the Pan-Tompkins algorithm [63, 66] to enhance the peak in the signal. First, we apply bandpass filter to the input signal to filter out noise. Next, the squaring operation is used to suppress the small values and enhance the large values. Then the output of squaring is filtered by a moving window integrator. Lastly, we detect heel strikes by searching for peaks, i.e., local maxima within a moving window.

Figure 4.7 demonstrates the process of stride detection based a 10-second data collected during running. We can see that the filters enhance the peaks associated with heel strikes, making the result of peak search more reliable and accurate.

Figure 4.8 shows the magnitude of overall acceleration, its projected vertical acceleration and its projected horizontal acceleration. The data is collected during running, and each spike is caused by a heel strike. We can see that, compared to the overall acceleration ($\| \vec{a} \|$ shown in Figure 4.8(a)), the projected vertical acceleration ($\| \vec{v} \|$ shown in Figure 4.8(b)) contains less noise and is more sensitive to the heel strikes caused by running.

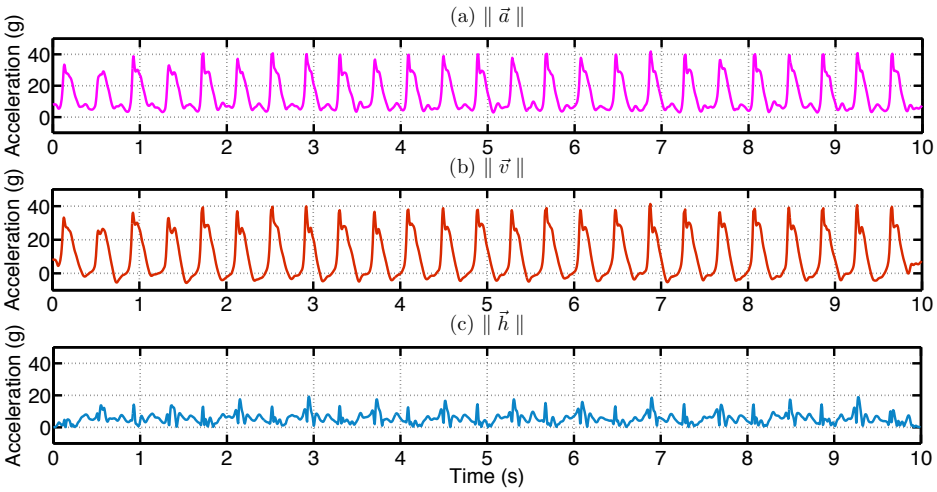


Figure 4.8 The result of projecting acceleration on to global coordinate. It is plotted based on a 10-second motion data collected when the subject is running with the phone in the armband. (a) is the magnitude of the raw acceleration reading. (b) and (c) show the magnitude of vertical and horizontal acceleration after projection, respectively.

4.4.5 LRC-based Correlation

The basic idea behind LRC-based correlation is to improve the accuracy of running rhythm measurement by estimating the most likely LRC ratio based on the correlation between detected stride and breathing. Since the motion sensor (accelerometer) is attached to user's body, the result of stride detection suffers far less from the environmental noise than that of breath detection. Specifically, there are two major factors that affect the performance of acoustic breath detection. One is the environmental noise, such as music in the gym, sound of wind or traffic, and etc. Another factor is the difficulty in detecting the breath, due to its low acoustic amplitude. Therefore, the estimated breath described in Breath Detection can not be directly used to calculate running rhythm.

Figure 4.9 shows an example of the breath detection result under the noise caused by passing traffic. The result is obtained using the algorithm described in Breath Detection. We can see that environment noise may cause both false alarms and low detection likelihood in breath detection. The false alarm is usually caused by noises that have similar MFCC features as the user's breath signature. For example, at 2.2 seconds, we can see the noise causes a false alarm with high likelihood, where there exists no actual breath event. Noise can also interfere with the sound of actual

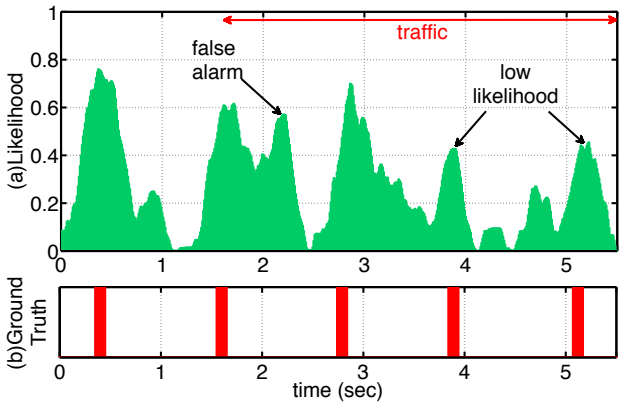


Figure 4.9 An example of inaccurate breath detection result under the noise caused by passing traffic.

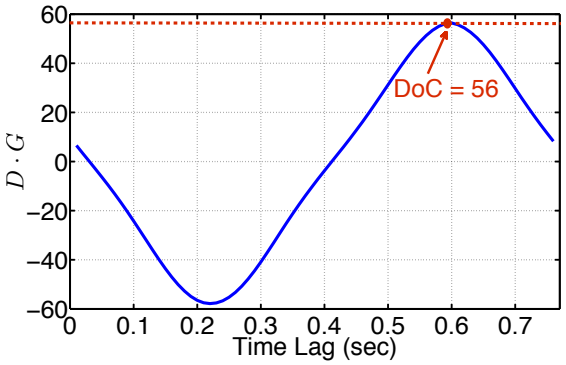


Figure 4.10 An example of computing Degree of Correlation (*DoC*) for 2:1 LRC ratio (using 10-second data). The arrow points to *DoC*, the maximum value of $D \cdot G$.

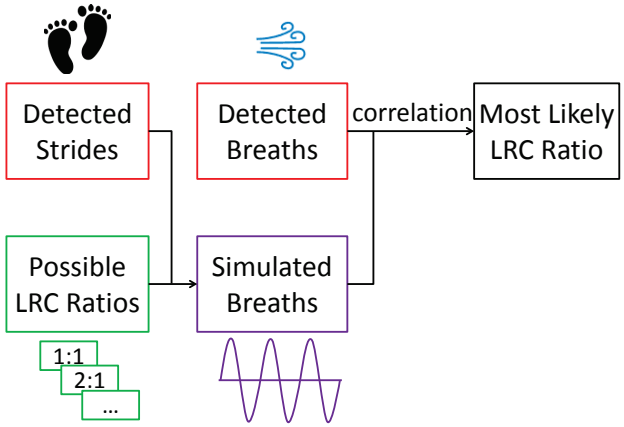


Figure 4.11 The process of LRC-based correlation. The detected breaths and strides are taken as inputs.

breath. Due to the interference, the MFCC features of the actual breath may largely deviate from the user's breath signature, resulting in low detection likelihood for actual breath. For example, the likelihood of the 4th and 5th breaths is only about half of that of the first three breaths.

According to the LRC theory, there exist several coupling ratios between the rhythm of stride and that of breath. The fundamental reason for the coupling effect is because breathing is responsible for supplying oxygen for metabolism, which keeps generating energy for locomotion. The design of LRC-based correlation in RunBuddy is based on two key observations. First, during running, breathing and stride frequencies are relatively stable within a short period of time (e.g.,

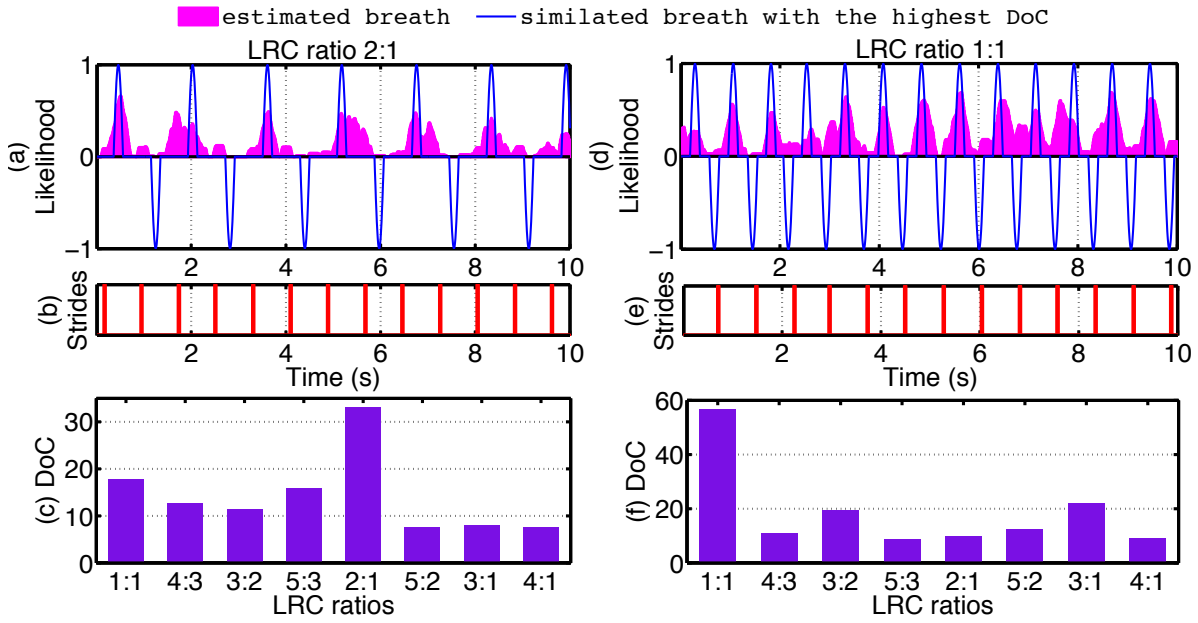


Figure 4.12 Two examples of LRC-based calibration process. The sine waves plotted in (a) and (d) are the simulated breath signals that have the highest correlation with detected breaths. (b) and (e) indicate the detected strides. (c) and (f) indicate *DoC*s associated with each LRC ratio.

10 seconds). Second, there only exists a limited number of LRC ratios in humans, and the coupling effect is stronger while running. This is because, as the intensity of the exercise gets higher (e.g., running), the rate of metabolism increases, resulting in a more obvious coupling effect. Therefore, RunBuddy is able to infer the accurate running rhythm using the most likely LRC ratio. To find the most likely LRC ratio, RunBuddy first use detected strides to generates breath signal, referred to as *simulated breath*, for each possible LRC ratio. RunBuddy then computes the correlation between the estimated breath and the simulated breaths. The LRC ratio that gives the highest degree of correlation is hence the mostly likely LRC ratio that represents the running rhythm.

Figure 4.11 shows the overview of the calibration process. First, we generate simulated breath signals using the detected strides for all 8 possible LRC ratios shown in Figure 4.2. Specifically, let T_{stride} be the duration of the stride cycle, and R_{LRC} be the ratio of stride frequency to breath frequency. The duration of simulated breath cycle T_{breath} can be calculated as follows,

$$T_{breath} = T_{stride} \times R_{LRC}; \tag{4.4}$$

We use sinusoid signal to represent the breath. The width of the sinusoid signal is fixed (e.g., 20 *ms*), regardless of the selected LRC ratio.

After generating the breath signals associated with different LRC ratios, we calculate the degree of correlation (DoC) between the simulated breaths and the detected breaths. Suppose $D = d_1, d_2, ..., d_n$ represents the estimated likelihood of breath for each frame within a short time window, $G(R) = g_1, g_2, ..., g_m$ is the corresponding breath signal simulated based on the strides and a particular LRC ratio R. The degree of correlation for LRC ratio R is given by,

$$\begin{aligned} DoC(R) &= \max_{s \in [1, T_{breath}]} \{D \cdot G(R, s)\} \\ &= \max\{\sum_{i=1}^{n} d_i \times g_{i+s}\} \end{aligned} \tag{4.5}$$

where s denotes the time lag of the simulated breath signal $G(R)$, and T_{breath} is the length of the breath cycle associated with $G(R)$. The basic idea behind the degree of correlation is based on cross-correlation, where the maximum result of the sliding dot product reflects the similarity of the detected breaths and simulated breaths. Figure 4.10 demonstrates an example of computing $DoC(R_{2:1})$ when LRC ratio is 2:1, and $G(R_{2:1})$ is generated based on the detected strides and 2:1 LRC ratio. We can see that the value of $D \cdot G(R_{2:1})$ varies as the time lag changes. And the maximum value is chosen as the $DoC(R_{2:1})$. In order to determine the most likely LRC ratio, we generate simulated breath signal for each LRC ratio (i.e., $G(R_{1:1})$, $G(R_{2:1})$,..., $G(R_{4:1})$), and calculate their corresponding *DoC*s (i.e., $DoC(R_{1:1})$, $DoC(R_{2:1})$,..., $DoC(R_{4:1})$). The LRC ratio that gives the highest *DoC* is chosen as the most likely LRC ratio that represents the current running rhythm. Moreover, since the LRC ratio defines the frequency ratio between stride and breath, we can obtain a more accurate breathing frequency than that derived from the estimated breath, using the most likely LRC ratio and the stride frequency (Equation 4.4).

In our implementation, we use a 10-second moving window with 50% overlap for the calibration. For example, Figure 4.12 shows the LRC-based calibration of two 10-second windows. The blue sine waves shown in Figure 4.12(a) and (d) indicate the simulated breaths associated with the

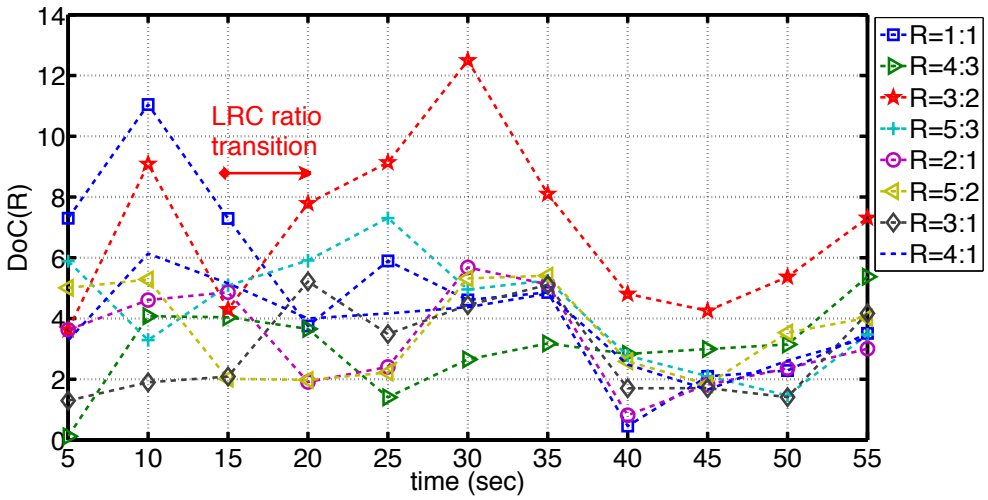


Figure 4.13 An example of degree of correlation (*DoC*) calculation for all possible LRC ratios where there exists a LRC ratio transition from 15 to 20 seconds.

most likely LRC ratio. We can observe that it gives the highest *DoC* among all LRC ratios (shown in Figure 4.12(a) and (d)). Moreover, we can see that it is difficult to accurately identify breaths by solely relying on the estimated breath likelihoods. Therefore, by cross-correlating the detected breaths and simulated breaths of different LRC ratios, we can effectively find the most likely LRC ratio, which can be used to represent the running rhythm.

Note that the LRC-based calibration needs to be repeated using a 10-second moving window because runners may switch to a different LRC ratio during running. According to our experimental result discussed in the following section, the frequency of LRC ratio change is highly dependent on each individual. In general, non-runners switch more frequently than regular runners [30, 16]. Figure 4.13 shows an example of calculated *DoC* of all possible LRC ratios over time, where the runner switched from LRC ratio 1:1 to 3:2 during 15 to 20 seconds. As mentioned above, RunBuddy calibrates the breath using a moving window of 10 seconds and 50% overlap, resulting in a *DoC* calculation every 5 seconds. Therefore, RunBuddy can effectively detect the LRC ratio switch during running.

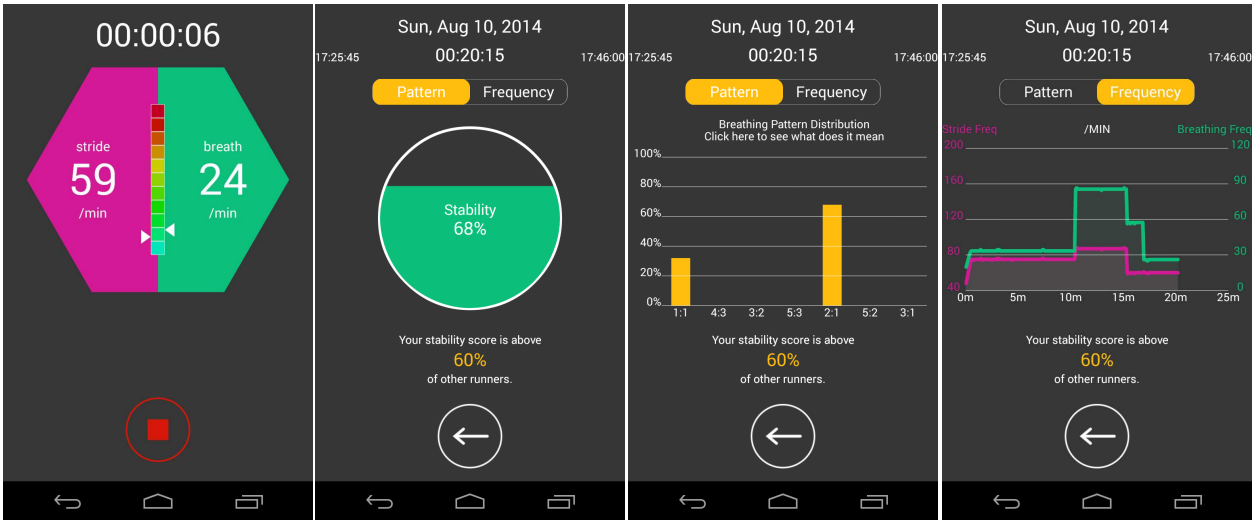
4.5 Implementation

RunBuddy is implemented on Android 4.3. The size of the application file is about 1 MB. While running, RunBuddy requires about 20 MB RAM allocation. The displaying and processing functions are implemented in separate threads, in order to ensure timely sampling and processing as well as quick response to user's interaction. RunBuddy continuously samples the microphone at 16 kHz and accelerometer at 100 Hz. The acoustic samples are framed into 100 ms frames before detection. The detection results of breathing and stride within a 10-second moving window with 50% overlap to measure LRC ratio, resulting in a measurement every 5 seconds.

For each frame, the system processes the acoustic samples with a low-pass filter, followed by the MFCC feature extraction. Then it calculates the likelihood of breath by comparing the extracted MFCC features and the user's breath signature from training data. To detect strides, the raw acceleration samples are first projected to global coordinate to obtain the vertical acceleration of the user. Then it processes the vertical acceleration using peak detection algorithm to detect strides. Finally, RunBuddy correlates the detection results of breathing and stride within a 10-second moving window with 50% overlap to measure the running rhythm, resulting in a measurement every 5 seconds.

We have released RunBuddy as a mobile App on the Google Play Store [3]. The screen shots are shown in Figure 4.14. The application is easy to use. After performing a one-time training, the user only needs to put on the bluetooth microphone and start the monitoring before running. During running, users are free to carry the phone in their preferable way (e.g., inside pocket or armband). As the app prevents the CPU from sleeping, the user is able to turn off the screen to save battery. It allows users to check their instant breathing and stride frequencies during running by simply turn on the screen. After finishing the running, the user needs to stop the monitoring to see an overall

[3]Our mobile app released on Google Play Store is branded as "iBreath". And the Google Play link is `https://play.google.com/store/apps/details?id=com.tian.ibreath&hl=en`. A short video introducing iBreath is at `http://youtu.be/hZMZqt4Pae4`. iBreath was awarded the Best Mobile App Award, Third Place, at MobiCom Mobile App Competition 2014. While iBreath has the identical design with RunBuddy, it presents the user breathing frequency (number of breaths per minute) that is derived from the measured LRC.



(a) (b) (c) (d)

Figure 4.14 The user interface of the app. (a) The screen showing the real-time breathing and stride frequency during running. (b) The screen showing the estimated fitness level according to the stability of running rhythm. (c) and (d) are the screens showing the details about the running rhythm distribution and the runner's breathing and stride frequency.

result reflecting the fitness level calculated based on the stability of his/her running rhythm. For each run, the app also provides detailed information including the distribution of running rhythm, as well as breathing and stride frequencies over time.

4.6 Evaluation

In this section, we evaluate the performance of RunBuddy with experiments. The results show that RunBuddy is able to effectively measure the breathing pattern continuously during running, across different users and environments. First, we introduce the experiment settings. Second, we describe the overall performance of RunBuddy. Next, we investigate the impact of different environments and users with different fitness levels. Lastly, we evaluate the computational overhead and power consumption of RunBuddy.

4.6.1 Experimental Settings

In order to evaluate the performance of RunBuddy, we recruited 13 subjects and collected data from 39 runs in total (526.1 minutes). Our study along with its data collection procedure was approved by the Institutional Review Boards (IRB) of the primary author's institute. All the subjects voluntarily agreed to help with the data collection, and signed a consent form. In order to collect data, each subject used a smartphone (Google Nexus 4 [58]) and a bluetooth headset during running. In order to investigate the impact of different bluetooth headsets, we used three different models (Jabra Wave [39], Voyager Legend [73] and Voyager Pro HD [74]) in our experiments. RunBuddy performs similarly with different bluetooth headsets.

During data collection, the acoustic data used for breath detection is collected through the bluetooth headset, and the acceleration data used for stride detection is collected using the phone's built-in accelerometer. The sampling frequency was empirically set to 16 kHz for the sound, and 100 Hz for the acceleration. The collected data was stored in the phone for off-line analysis. The ground truth for breath detection is mainly collected by attaching an in-line microphone under the subject's nose. In the cases where the subject felt uncomfortable to wear the microphone under nose, or the microphone accidentally fell off during running, we manually labeled the breath events to get the ground truth.

During data collection, the acoustic data used for breath detection is collected through the bluetooth headset, and the acceleration data used for stride detection is collected using the phone's built-in accelerometer. The sampling frequency was empirically set to 16 kHz for the sound, and 100 Hz for the acceleration. The collected data was stored in the phone for off-line analysis. The ground truth for breath detection is collected by either of the following two methods. In the first method, an in-line microphone is attached under the subject's nose during running. Since the in-line mic is very close to the subject's nose and mouth, the recorded breathing sound is clear enough to be automatically labeled offline through simple processing. In the second method, we manually labeled the breath event if the subject felt uncomfortable to wear the in-line mic under nose, or the in-line mic accidentally fell off during running. Specifically, the labeling is done by listening to

category	subject	sex	# of runs	avg. duration	BMI
Non-runner	1	F	1	6.1 min	16.9
Non-runner	2	F	1	5.8 min	17.9
Non-runner	3	M	2	5.1 min	27.5
Non-runner	4	F	3	5.2 min	18.3
Non-runner	5	M	1	8.3 min	28.4
Occasional	6	M	7	6.3 min	25.4
Occasional	7	M	2	16.1 min	27.2
Occasional	8	M	3	12.1 min	24.4
Occasional	9	M	3	14.5 min	22.1
Occasional	10	F	3	13.8 min	20.5
Occasional	11	M	4	12.9 min	23.1
Occasional	12	F	3	15.6 min	19.1
Regular	13	M	6	30.7 min	22.4

Table 4.1 General information about the subjects.

the audio recorded by the bluetooth headset and finding the time of the beginning of each breath event. The recorded audio clips where the breathing sound is not clear enough for manual labeling were discarded and not used in the evaluation. To collect the ground truth for stride detection, an extra smartphone was attached to the user's lower leg using an armband.

Table 4.1 shows the general information about the subjects who participated in our experiments. In order to assess their capability of aerobic exercise, we asked the subjects to fill out a questionnaire before the experiment, which includes several questions such as "How many days per week do you exercise?". Based on their self-report information, we divided the subjects into three categories as follows: **Non-runner**, subjects who rarely take any forms of exercise; **Occasional runner**, subjects who occasionally take short aerobic exercise, but do not have a regular routine; **Regular runner**, subjects who usually take more than 30 minutes of exercise for at least twice a week. To assess the subjects' body shape, we also calculated their body mass index (BMI) based on their weight and height. According to the standard specified by the World Health Organization (WHO) [18], the body shape can be classified as follows: **Underweight** ($BMI \leq 18.5$), **Normal weight** ($18.5 < BMI \leq 24.9$), **Overweight** ($25 < BMI \leq 29.9$) and **Obesity**($BMI \geq 30$). As shown in Table 4.1, all three female subjects who are non-runners were classified as under-

weight, while the BMIs of the other two female subjects who are occasional runners are in the lower range of normal weight. The two male subjects who are non-runners have much higher BMIs (27.5 and 28.4) than the five male subjects who are occasional runners (avg. BMI = 24.4) and the male regular runner, whose BMI is 22.4. This observation is consistent with the fact that exercise is effective in helping avoid overweight and maintain good body shape.

Prior to the experiment, each subject was instructed to run as they normally do. We did not set a specific duration for the run, and the subjects were asked to keep running as long as they felt comfortable. Note that the subject is free to choose where to place the smartphone during running. In our experiments, the ways of carrying the phone include holding in one hand, placing inside the pants' pocket, and attaching to the arm using sport armband. Our result shows that, regardless of the phone placement, RunBuddy can reliably detect strides with an accuracy of over 99%.

4.6.2 Metrics

We processed the collected data using the methods described in System Design. The metric we use to evaluate the performance of RunBuddy is the *percentage of correct measurements* (PCM). PCM = $X\%$ means RunBuddy correctly provides the LRC ratio measurement $X\%$ of the time during the run. For instance, for a run about 8 minutes consisting of 100 measurement windows, 95% PCM indicates 95 measurements are correct. The reason that we use PCM instead of accuracy or error rate as the metric is because the measurement of running rhythm in terms of LRC ratio is represented by discrete values. Also, we note that in most false measurements, RunBuddy produced a LRC ratio close to the ground truth (e.g., 4:3 while the correct ratio is 3:2).

4.6.3 Overall Performance

In this section, we evaluate RunBuddy when the users run on the treadmill in typical gyms and outdoors. Figure 4.15(a) shows the average PMC for each subject based on data collected during 29 indoor runs. The overall PMC for the indoor runs is 93.3%, with the lowest PMC of 87.3% for subject 1, and the highest PMC of 95.2% for subject 6. We can make two interesting observations

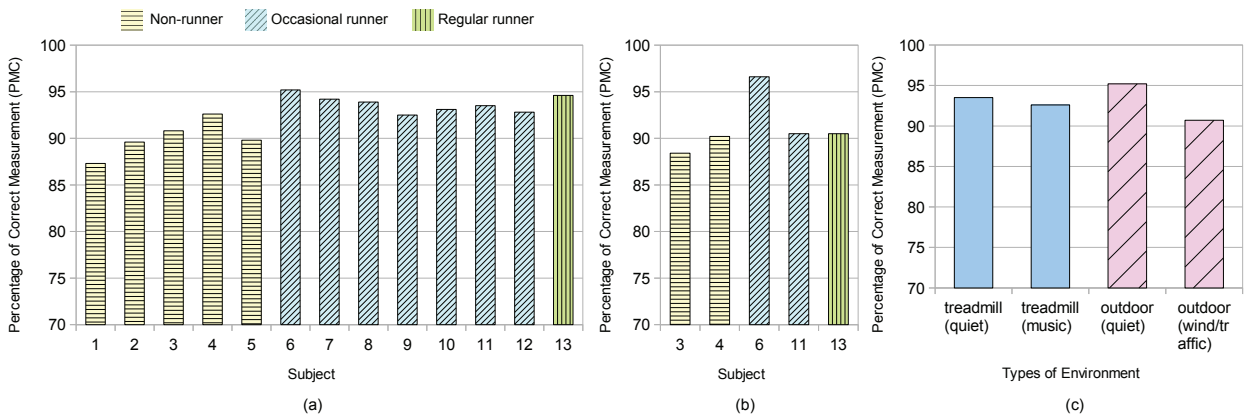


Figure 4.15 (a) shows the PCM for each subject based on data collected on treadmill in typical gyms. (b) shows the PCM for each subject based on data collected during outdoor runs. (c) shows the overall PCM in 4 typical scenarios: (1) running on treadmill in a relatively quiet gym, (2) running on treadmill in a gym with music and nearby runners, (3) running outdoors on a relatively quite route, (4) running outdoors by the street with traffic.

from the indoor result. First, RunBuddy tends to yield lower PMC for data collected from female subjects (subject 1, 2, 4 10 and 12). This is primarily because the sound of breathing of females typically have lower intensity than that of males, making it more difficult to be recognized. Another observation is that the accuracy for all five subjects who are non-runners (subject 1, 2, 3, 4 and 5) are below PMC (93.3%). The major reason is that subjects of lower fitness level usually take shorter and shallower breath during running. However, it is widely recommended by professional runners and running couches to take long and deep breath during running, which helps to obtain enough oxygen and prevent muscle and lung fatigue [25].

Figure 4.15(b) shows the PMC for each subject based on data collected during 10 outdoor runs. One key observation is that, for all subjects except subject 6, the average PMC of outdoor runs is higher than that of indoor runs, due to the impact of environment noises like wind and traffic. As an exception, all the 3 outdoor runs of subject 6 were collected in late evening in a quite neighborhood.

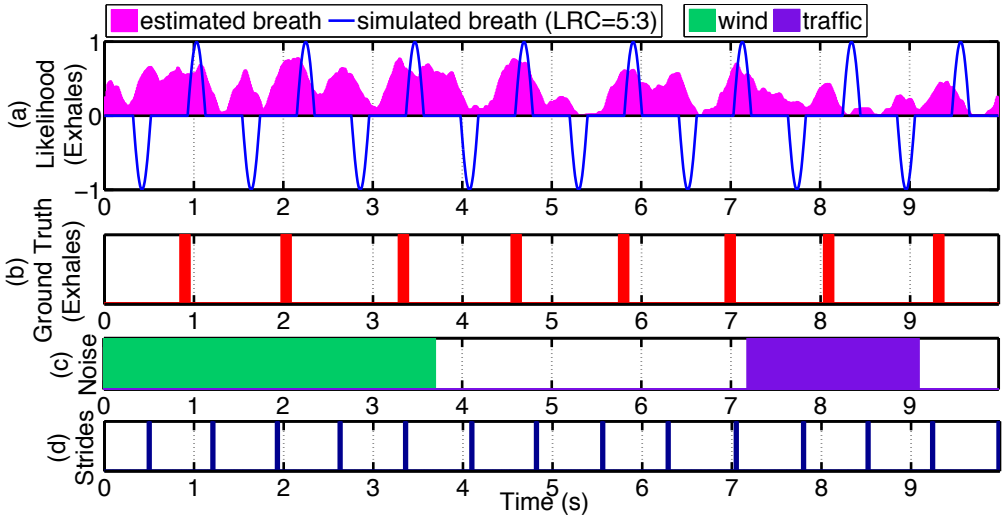


Figure 4.16 A 10-second example based on real data collected outdoors, demonstrating the impact of environment noises such as wind and traffic. (a) shows the estimated breath and simulated breath with the highest *DoC*. (b) is the ground truth for breath. (c) shows the time and duration of environment noises. (d) is the detection result of stride.

4.6.4 Impact of Environmental Noises

Next, we investigate the performance of RunBuddy in different environments. Figure 4.15(c) shows the PMC in 4 different scenarios based on all the data (39 runs). We can see that compared to outdoor running (avg. PMC 91.4%), the PMC is higher when running on indoor treadmill (avg. PMC 93.3%). This discrepancy is largely due to the outdoor noises, such as wind and traffic. However, we note that the highest PMC (95.2%) is achieved when running outdoors in a quite neighborhood, even 1.7% higher than running in a quiet gym. This is largely due to the fact that operating treadmills still produce substantial noise. In a typical public gym setting, the PMC drops to 92.6%, due to the raised noise level caused by nearby runners and ambient music. When running outside, wind is the most common environment noise, especially when running against the wind. As the noise of wind has overlapping frequency with the sound of breath, it can largely affect the accuracy of breath detection.

However, in most cases, RunBuddy can still manage to accurately estimate the running rhythm by leveraging the LRC-based correlation, even in the presence of wind and traffic. Figure 4.16 shows an example of how RunBuddy mitigates the impact of environmental noises. Due to the

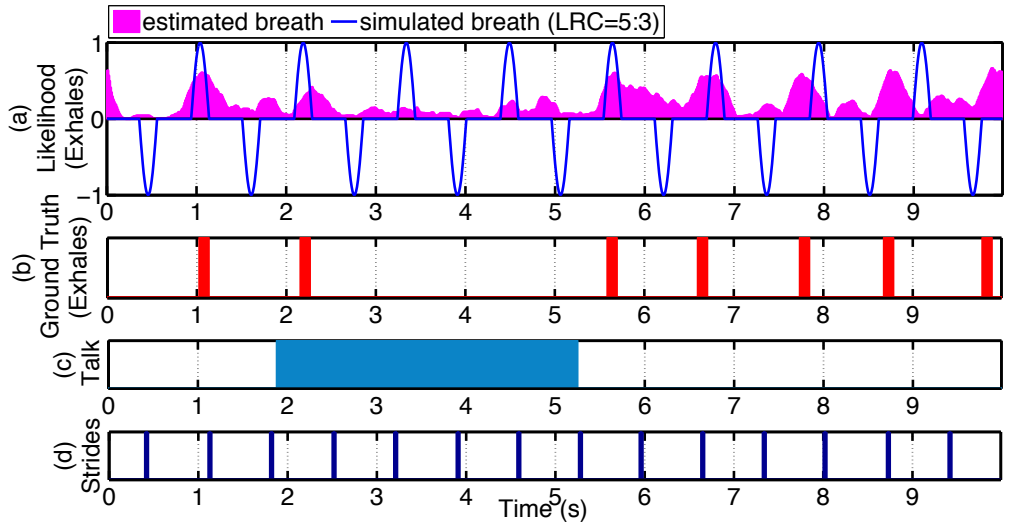


Figure 4.17 A 10-second example based on real data collected outdoors, demonstrating the impact of talking during running. (a) shows the estimated breath and simulated breath with the highest *DoC*. (b) is the ground truth for breath. (c) shows the time and duration when the runner was talking. (d) is the detection result of stride.

interference of the wind, the breath detection algorithm yields continuous high likelihood of breath during the first 3.7 seconds. During 7.2 to 9.1 seconds, a vehicle was passing by the subject. We can see that, in detecting the 7th and 8th breaths, the traffic noise lowers the likelihood of breath by suppressing the sound of breath. An interesting observation is that, even though some of the detected breath do not match the breathes in ground truth, the number of detected breathes is still accurate, due to the fact that the LRC ratio remains constant and the strides are detected accurately. Therefore, by leveraging the LRC model, RunBuddy can largely mitigate the impact of various outdoor noises, including wind and nearby traffic.

Figure 4.17 shows another example illustrating the impact of talking. As shown in Figure 4.17(c), the subject briefly talked to his running partner from 1.9 to 5.2 seconds. From the ground truth (Figure 4.17(b)) we can observe that the subject held breath for two breath cycles during the talking. Moreover, the second breath (at 2.2 seconds) occurred at the beginning of the talking is also affected, resulting in a shorter and shallower breath. Another interesting observation is that, after the subject finished talking, two deeper and longer breaths (at 5.6 and 6.6 seconds) are detected, followed by three faster breathes. This is consistent with the fact that the subject needed more

oxygen through stronger and quicker respiration to compensate the held breathes. As shown in Figure 4.17(a), the peaks of estimated breathes (at 7.8 and 8.7 seconds) are largely shifted from those of the simulated breathes, due to the irregular breath frequency after talking. In general, talking is not common during running and hence has limited impact on the performance of RunBuddy. During our experiments, talking only takes about 10% of the total time of running. In addition, as shown by our results, talking often leads to irregular breathing patterns, and the breath measurement during talking has little value to the users. RunBuddy may integrate voice detection algorithms [65] to remove the talking periods from the measurement result.

4.6.5 Runners with Different Fitness Levels

RunBuddy is designed to help improve the running experience of both regular runner and non-runners alike. As regular runners typically breathe differently from non-runners, in terms of depth and duration, it is important to evaluate the performance of RunBuddy for runners at different fitness levels. According to the result of our initial experiment, the PMC of LRC ratio measurement increases as the fitness level of the subject gets higher. Specifically, the average PMC of indoor running for non-runners, occasional runners and regular runner are 90%, 93.6% and 94.6%, respectively. This is primarily due to the fact that regular runners typically breathe deeper and longer, resulting in more distinctive sound features that are easier for the system to capture.

In addition to the depth and duration, the breathing of regular runners and non-runners also differs in its coupling with the strides. Figure 4.18 shows the detailed detection results of a 17-min run from a regular runner, and an 8-min run from a non-runner. As shown in Figure 4.18(a), we can see that the subject gradually raised his breathing and stride frequencies in the first minute. Then, he maintained a steady stride frequency till 14:30 minutes. After that, the subject gradually reduced the speed and switched from running to fast walking. One interesting observation is that the subject doubled his breathing frequency for about 1 minute (around 14 minutes) at the end of the running. It is mainly because he raised the speed of running by increasing the length of each stride. As a result, he maintained the same stride frequency while increasing the breathing frequency, which

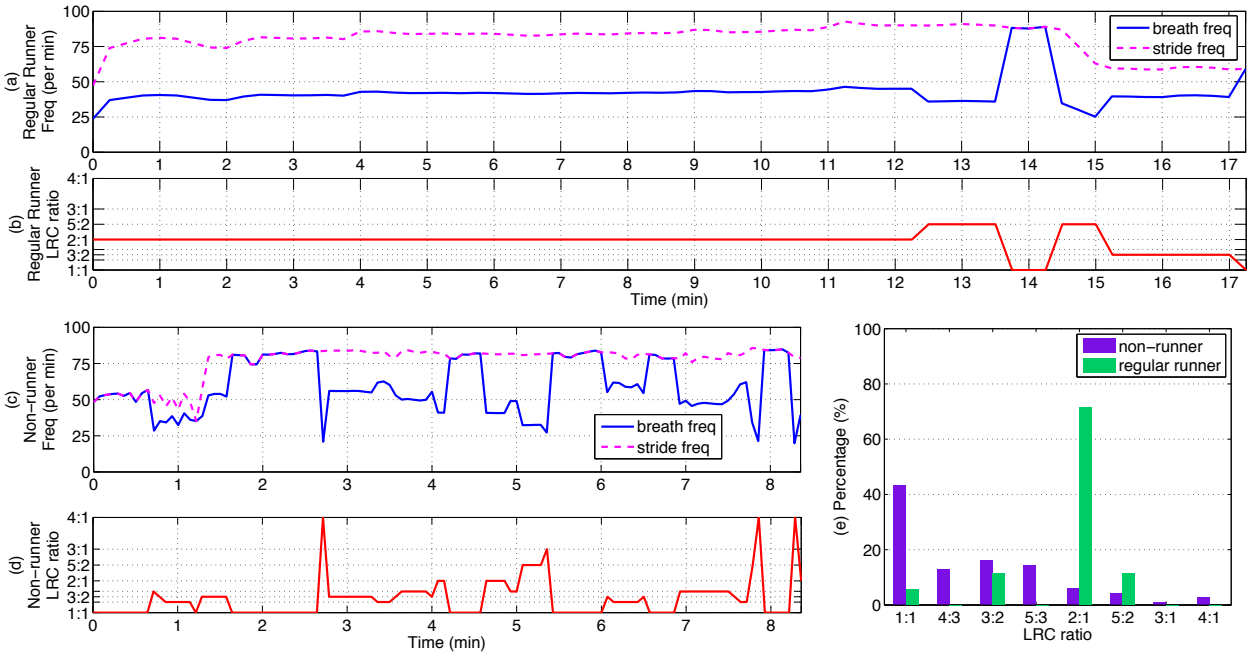


Figure 4.18 The detection result of typical runs from a regular runner (17.2 min, Category 2) and a non-runner (8.4 min, Category 1). (a) and (c) show the frequency of breath and stride for regular runner and non-runner, respectively. (b) and (d) show their LRC ratios over time. (e) shows the distributions of their LRC ratio used during running.

can also be reflected in Figure 4.18(b) as the corresponding LRC ratio shifts to 1 : 1. Therefore, RunBuddy is able to provide details of the dynamics in stride/breath frequencies to regular runners, which help them better understand their running progress. It is worth mentioning that one of our subjects who runs regularly is very enthusiastic about the breathing pattern results provided by RunBuddy. He thought the detection result of RunBuddy is a valuable feedback, which could help even regular runners “push that extra little bit".

As shown in Figure 4.18(c), it took the non-runner one and a half minutes to reach a relatively steady stride frequency. However, it can be seen that the subject kept changing the breathing frequency. As reported by the subject, she found it difficult to adjust her breath to her stride during running. Actually, the shortly dropped breathing frequency at around 2:45 and 7:50 minutes reflect her effort in trying to adjust her breath frequency. However, due to the lack of running experience and relatively low fitness level, she failed to find a comfortable running rhythm and kept adjusting the breathing frequency throughout the run. This also leads to a varying LRC ratio over time.

Therefore, by continuously monitoring the running rhythm along with other information (e.g., stride), RunBuddy has potential to guide the non-runners to find a comfortable running rhythm (e.g. play music with proper tempo based on the analysis of the LRC measurements)

Figure 4.18(e) shows the percentage of each LRC ratio of the non-runner and the regular runner. We can see that the dominant LRC ratio (2:1) of the regular runner was used for around 70% of the run, whereas the non-runner used 8 different LRC ratios, with the dominant LRC ratio 1:1 being used for only 43% of the time. This result indicates that the coupling effect between breathing and stride of the regular runner is stronger than that of the non-runner. This observation is consistent with the literature of physiology [20], suggesting that tighter coordination between limb rhythms and respiration may reduce the metabolic cost of a movement.

4.6.6 Computation Overhead and Power Consumption

In this section, we evaluate the overhead of RunBuddy implemented on Android platform (as discussed in Section 4.5). Here we focus on the the evaluation of two major pipelines, which are, *audio pipeline*, including the low-pass filtering, MFCC feature extraction and breath detection, and *motion pipeline* consisting of vertical projection and stride detection components. In the audio pipeline, the most computation is caused by the MFCC feature extraction, followed by the low-pass filtering. Compared to the other two components, the computational cost of breath detection is significantly lower. This is mainly because the size of input data stream is largely reduced from $80,000$ to 50 every 5 seconds after MFCC feature extraction. In addition, RunBuddy further reduces the computational overhead by adopting a simple yet efficient algorithm based on calculating the similarity with the training data.

We evaluated the computation overhead of RunBuddy on 4 different platforms, including Samsung Galaxy Nexus [32], Motorola Moto G [57], LG Nexus 4 [58] and LG Nexus 5 [59]. Specifically, we measured the runtime for each pipeline to process 5-second data. The result is shown in Table 4.2. We can see that the runtime of the audio pipeline is significantly more than that of the motion pipeline. For example, the time taken by the motion pipeline in Nexus 4 is only about

Model	CPU	RAM	Battery	avg/max load	RT(audio)	RT(motion)	Pwr C.
Galaxy N.	Dual 1.2GHz	1GB	1750 mAh	4.7%/7%	595ms/5sec	3.4ms/5sec	5%/hr
Moto G	Quad 1.2GHz	1GB	2070 mAh	3.5%/4%	560ms/5sec	3.0ms/5sec	5%/hr
Nexus 4	Quad 1.5GHz	2GB	2100 mAh	8.2%/14%	522ms/5sec	2.8ms/5sec	5%/hr
Nexus 5	Quad 2.3GHz	2GB	2300 mAh	5.2%/8%	318ms/5sec	2.2ms/5sec	4%/hr

Table 4.2 The overhead of RunBuddy on different smartphones. The computational overhead is measured by the overall CPU load of RunBuddy and the time consumed for each pipeline to process 5-second data. Power consumption is measured by the battery usage per hour.

0.5% of that of the audio pipeline. Another observation is that both pipelines take less time on the platform with faster CPU and more RAM. Overall, the results show that RunBuddy takes a small amount of processing time because of the lightweight design of sensing pipelines. We also measured the CPU load of RunBuddy by using the Android Debug Bridge (adb) provided in the Android SDK. The average and max CPU loads of RunBuddy are shown in Table 4.2. We can see that, the average CPU loads in all platforms are below 10%, while the maximum CPU load is below 15%.

Next, we evaluate the power consumption of RunBuddy. First, we record the phone's remaining battery level $r_1 \in [0,1]$. Then we start running RunBuddy on the phone and turn off the screen. Lastly, after an hour, we exit RunBuddy and record the remaining battery level $r_2 \in [0,1]$. The hourly battery usage of RunBuddy is estimated as $r_2 - r_1$. Note that the estimation result gives an upper bound of RunBuddy's battery usage, since we do not exclude the power consumption of system services. The result of the battery usages of 4 different phone is shown in Table 4.2. We can see that RunBuddy consumes around 5% of battery on platforms with various battery capacities.

4.7 Summary

In this chapter, we present RunBuddy – the first smartphone-based system for continuous running rhythm monitoring. RunBuddy only utilizes commodity devices including smartphone and bluetooth headset, and is convenient and unobtrusive to users. RunBuddy adopts lightweight and efficient signal processing algorithms for detecting breathes and strides using accelerometer and

microphone, respectively. It then leverages a physiological model called Locomotor Respiratory Coupling (LRC) to correlate the sensing results, which significantly improves the performance. RunBuddy is evaluated through extensive experiments involving 13 subjects and 39 runs. Our results show that RunBuddy can accurately measure the running rhythm in terms of LRC ratio in 92.7% of the time. We examine the performance of RunBuddy in various indoor and outdoor environments. By leveraging LRC models, RunBuddy yields robust performance even in the presence of various environmental noises. RunBuddy is able to provide physiological details of running (e.g., the stability of LRC ratio) that can be used to help users to better understand the running process and improve the running experience. Moreover, the real-time running rhythm measurement provided by RunBuddy also has great potential in helping runners better coordinate their breathing and strides, and improve their running performance.

4.8 Disclaimer

We have obtained IRB approval (IRB# 13-791) from the Institutional Review Board at Michigan State University for the use of human subjects in this study.

CHAPTER 5

CONCLUSION

The idea of continuously monitoring physiological events is gaining popularity, because of its potential contribution in disease prevention and diagnosis. By measuring and analyzing long-term physiological events, the physical and mental health of the user can also be inferred. As a device that is carried by the user, smartphone is considered as the ideal platform to perform long-term and continuous monitoring of physiological events. However, there are still a number of technical barriers to tackle in order to make these physiological monitoring applications truly benefit the end uses. In this thesis, we have identified the requirements and challenges in developing smartphone-based physiological monitoring applications, and provided design solutions to deal with these challenges with respect of specific applications.

The primary contribution of this thesis is two-fold. First, we demonstrate the feasibility of smartphone-based physiological sensing by focusing on a specific health dimension, which is sleep. We present the design, implementation, and evaluation of iSleep – a practical system to monitor an individual's sleep quality using off-the-shelf smartphone. The design of iSleep faces several challenges such as highly diverse acoustic profiles of sleep from person to person and in different environments. We carefully analyze the acoustic data collected from real sleep experiments and choose several statistical acoustic features that can differentiate environment noise and various sleep-related events. To improve the robustness of detection, iSleep tracks the ambient noise characteristics and updates the noise model adaptively. We have evaluated iSleep extensively in a long-term experiment that involves 7 participants and total 51 nights of sleep, as well as using the data collected from the Android phones that downloaded and installed iSleep from Google Play Store (MSU IRB#12-1178). Our results show that iSleep achieves consistently above 90% classification accuracy for various events, across different subjects and in a variety of different sleep environments.

Second, we propose a novel idea that helps improve the mobile sensing accuracy by leveraging physiological model. We further demonstrate the integration of mobile sensing and physiological model by presenting the design of RunBuddy – a smartphone-based system for running rhythm monitoring. In order to improve the sensing accuracy, we propose a novel approach in the design of RunBuddy, which integrates ambient sensing based on accelerometer and microphone and a physiological model called Locomotor Respiratory Coupling (LRC) – a quantitative measure of the coordination of locomotory and respiratory systems. RunBuddy is evaluated through extensive experiments involving 13 subjects and 39 runs (MSU IRB#13-791). Our results show that RunBuddy can accurately measure the running rhythm in terms of LRC ratio in 92.7% of the time.

APPENDIX

APPENDIX

PUBLICATIONS AND AWARDS

1 Conference Publications

- **Tian Hao**, Guoliang Xing, and Gang Zhou. "*iSleep: unobtrusive sleep quality monitoring using smartphones*." Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys). ACM, 2013.

- **Tian Hao**, Ruogu Zhou, and Guoliang Xing. "*Cobra: color barcode streaming for smartphone systems*." Proceedings of the 10th international conference on Mobile systems, applications, and services (MobiSys). ACM, 2012.

- **Tian Hao**, Ruogu Zhou, Guoliang Xing, Matt W. Mutka. "*Wizsync: Exploiting wi-fi infrastructure for clock synchronization in wireless sensor networks*." Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd. IEEE, 2011.

2 Journal Publications

- **Tian Hao**, Ruogu Zhou, Guoliang Xing, Matt W. Mutka, Jiming Chen."*WizSync: Exploiting Wi-Fi Infrastructure for Clock Synchronization in Wireless Sensor Networks*", IEEE Transactions on Mobile Computing, , no. 1, pp. 1, PrePrints PrePrints, doi:10.1109/TMC.2013.43

3 Awards

- **MobiCom 2013 Best Mobile App Award**, *Third Place for iSleep:unobtrusive sleep monitoring system*, The 19th ACM Annual International Conference on Mobile Computing and Networking (MobiCom), September, 2013, Miami, Florida.

- **MobiCom 2014 Best Mobile App Award**, *Third Place for iBreath: breathing monitoring system during running*, The 20th ACM Annual International Conference on Mobile Computing and Networking (MobiCom), September, 2014, Maui, Hawaii.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] "Fitbit," http://www.fitbit.com/.

[2] "https://play.google.com/store/apps/."

[3] "Sleep as android," https://play.google.com/store/apps/details?id=com.urbandroid.sleep.

[4] "Sleep as android," https://sites.google.com/site/sleepasandroid/.

[5] "Sleep cycle," http://www.sleepcycle.com/.

[6] "Sleep cycle," http://www.sleepcycle.com/.

[7] "Sleep tracker," http://www.sleeptracker.com/.

[8] "Watch pat," http://www.itamar-medical.com/WatchPAT.html.

[9] "Zeo," http://www.myzeo.com/sleep/.

[10] E. Aaron, K. Seow, B. Johnson, and J. Dempsey, "Oxygen cost of exercise hyperpnea: implications for performance," *Journal of Applied Physiology*, vol. 72, no. 5, pp. 1818–1825, 1992.

[11] S. Ancoli-Israel *et al.*, "The role of actigraphy in the study of sleep and circadian rhythms," in *Sleep*, vol. 26, no. 3, 2003, pp. 342–392.

[12] M. H. Anshel and D. Q. Marisi, "Effect of music and rhythm on physical performance," *Research Quarterly. American Alliance for Health, Physical Education and Recreation*, vol. 49, no. 2, pp. 109–113, 1978.

[13] J. Backhaus *et al.*, "Test-retest reliability and validity of the pittsburgh sleep quality index in primary insomnia," *Journal of psychosomatic research*, vol. 53, no. 3, pp. 737–740, 2002.

[14] Y. Bai, B. Xu, Y. Ma, G. Sun, and Y. Zhao, "Will you have a good sleep tonight? sleep quality prediction with mobile phone," in *BodyNets*, 2012.

[15] P. Bernasconi, P. Bürki, A. Bührer, E. Koller, and J. Kohl, "Running training and co-ordination between breathing and running rhythms during aerobic and anaerobic conditions in humans," *European journal of applied physiology and occupational physiology*, vol. 70, no. 5, pp. 387–393, 1995.

[16] P. Bernasconi and J. Kohl, "Analysis of co-ordination between breathing and exercise rhythms in man." *The Journal of physiology*, vol. 471, no. 1, pp. 693–706, 1993.

[17] J. T. Biehl, P. D. Adamczyk, and B. P. Bailey, "Djogger: a mobile dynamic music device," in *CHI'06 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2006, pp. 556–561.

[18] "Body Mass Index," http://apps.who.int/bmi.

[19] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013, pp. 225–234.

[20] D. M. Bramble and D. R. Carrier, "Running and breathing in mammals," *Science*, vol. 219, no. 4582, pp. 251–256, 1983.

[21] D. Buysse *et al.*, "Quantification of subjective sleep quality in healthy elderly men and women using the pittsburgh sleep quality index (psqi)." *Sleep*, vol. 14, no. 4, pp. 331–338, 1991.

[22] D. Buysse, C. Reynolds, T. Monk, S. Berman, and D. Kupfer, "The pittsburgh sleep quality index (psqi): A new instrument for psychiatric research and practice," in *Psychiatry Research*, vol. 28, no. 2, 1989, pp. 193–213.

[23] A. Chesson *et al.*, "Practice parameters for the indications for polysomnography and related procedures," in *Sleep*, vol. 20, no. 6, 1997, pp. 406–422.

[24] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao, "Automatically characterizing places with opportunistic crowdsensing using smartphones," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 481–490.

[25] B. Coates and C. Kowalchik, *Runner's World Running on Air: The Revolutionary Way to Run Better by Breathing Smarter*. Rodale, 2013.

[26] R. Cole *et al.*, "Technical note automatic sleep/wake identification from wrist activity," in *Sleep*, vol. 15, no. 5, 1992, pp. 461–469.

[27] R. De Oliveira and N. Oliver, "Triplebeat: enhancing exercise performance with persuasion," in *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*. ACM, 2008, pp. 255–264.

[28] K. R. Fox, "The influence of physical activity on mental well-being," *Public health nutrition*, vol. 2, no. 3a, pp. 411–418, 1999.

[29] A. Ganapathiraju, J. Hamaker, J. Picone, and A. Skjellum, "A comparative analysis of fft algorithms," *the IEEE Transactions on Signal Processing*, 1997.

[30] F. Garlando, J. Kohl, E. Koller, and P. Pietsch, "Effect of coupling the breathing-and cycling rhythms on oxygen uptake during bicycle ergometry," *European journal of applied physiology and occupational physiology*, vol. 54, no. 5, pp. 497–501, 1985.

[31] L. K. George, D. G. Blazer, D. C. Hughes, and N. Fowler, "Social support and the outcome of major depression." *The British Journal of Psychiatry*, vol. 154, no. 4, pp. 478–485, 1989.

[32] "Samsung Galaxy Nexus," http://www.samsung.com/us/mobile/cell-phones/SPH-L700ZKASPR.

[33] J. Hedner *et al.*, "A novel adaptive wrist actigraphy algorithm for sleep-wake assessment in sleep apnea patients," in *Sleep*, vol. 27, no. 8, 2004, pp. 1560–1566.

[34] C. P. Hoffmann, G. Torregrosa, and B. G. Bardy, "Sound stabilizes locomotor-respiratory coupling and reduces energy cost," *PloS one*, vol. 7, no. 9, p. e45206, 2012.

[35] E. Hoque, R. Dickerson, and J. Stankovic, "Monitoring body positions and movements during sleep using wisps," in *Wireless Health*, 2010.

[36] C. Iber *et al.*, "The aasm manual for the scoring of sleep and associated events," in *Westchester, IL: American Academy of Sleep Medicine*, 2007.

[37] S. Imai, "Cepstral analysis synthesis on the mel frequency scale," in *ICASSP*, vol. 8, 1983, pp. 93–96.

[38] T. Iso and K. Yamazaki, "Gait analyzer based on a cell phone with a single three-axis accelerometer," in *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*. ACM, 2006, pp. 141–144.

[39] "Jabra Wave Mobile Bluetooth Headset," http://www.jabra.com/products/bluetooth/jabra_wave/jabra_wave.

[40] C. Karageorghis, L. Jones, and D. Stuart, "Psychological effects of music tempi during exercise," 2007.

[41] M. Kay, E. K. Choe, J. Shepherd, B. Greenstein, N. Watson, S. Consolvo, and J. A. Kientz, "Lullaby: a capture & access system for understanding the sleep environment," in *UbiComp*, 2012.

[42] J. Kohl, E. Koller, and M. Jäger, "Relation between pedalling-and breathing rhythm," *European journal of applied physiology and occupational physiology*, vol. 47, no. 3, pp. 223–237, 1981.

[43] E. C. Larson, M. Goel, G. Boriello, S. Heltshe, M. Rosenfeld, and S. N. Patel, "Spirosmart: Using a microphone to measure lung function on a mobile phone," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 280–289.

[44] E. C. Larson, T. Lee, S. Liu, M. Rosenfeld, and S. N. Patel, "Accurate and privacy preserving cough sensing using a low-cost microphone," in *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 2011, pp. 375–384.

[45] J. Lester, C. Hartung, L. Pina, R. Libby, G. Borriello, and G. Duncan, "Validated caloric expenditure estimation using a single body-worn sensor," in *Proceedings of the 11th international conference on Ubiquitous computing*. ACM, 2009, pp. 225–234.

[46] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, "A reliable and accurate indoor localization method using phone inertial sensors," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 421–430.

[47] J. Liu, W. Xu, M.-C. Huang, N. Alshurafa, and M. Sarrafzadeh, "A dense pressure sensitive bedsheet design for unobtrusive sleep posture monitoring," in *PerCom*, 2013.

[48] S. H. Loring, J. Mead, and T. B. Waggener, "Determinants of breathing frequency during walking," *Respiration physiology*, vol. 82, no. 2, pp. 177–188, 1990.

[49] H. Lu, W. Pan, N. Lane, T. Choudhury, and A. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in *MobiSys*, 2009.

[50] H. Lu, M. Rabbi, G. Chittaranjan, D. Frauendorfer, M. Mast, A. Campbell, D. Gatica-Perez, and T. Choudhury, "Stresssense: Detecting stress in unconstrained acoustic environments using smartphones," in *UbiComp*, 2012.

[51] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The jigsaw continuous sensing engine for mobile phone applications," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2010, pp. 71–84.

[52] D. A. Mahler, B. Hunter, T. Lentine, and J. Ward, "Locomotor-respiratory coupling develops in novice female rowers with training." *Medicine and science in sports and exercise*, vol. 23, no. 12, pp. 1362–1366, 1991.

[53] D. A. Mahler, C. R. Shuhart, E. Brew, and T. A. Stukel, "Ventilatory responses and entrainment of breathing during rowing." *Medicine and science in sports and exercise*, vol. 23, no. 2, pp. 186–192, 1991.

[54] W. J. McDermott, R. E. Van Emmerik, and J. Hamill, "Running training and adaptive strategies of locomotor-respiratory coordination," *European journal of applied physiology*, vol. 89, no. 5, pp. 435–444, 2003.

[55] T. Mitchell, *Machine Learning*. McGraw-Hill.

[56] D. Mizell, "Using gravity to estimate accelerometer orientation," in *2012 16th International Symposium on Wearable Computers*. IEEE Computer Society, 2003, pp. 252–252.

[57] "Motorola Moto G," http://www.motorola.com/us/consumers/moto-g/Moto-G/moto-g-pdp.html.

[58] "Nexus 4 (android smartphone)," http://www.google.com/intl/ALL/nexus/4/.

[59] "LG Nexus 5," https://www.google.com/nexus/5/.

[60] S. Nirjon, R. F. Dickerson, Q. Li, P. Asare, J. A. Stankovic, D. Hong, B. Zhang, X. Jiang, G. Shen, and F. Zhao, "Musicalheart: A hearty way of listening to music," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM, 2012, pp. 43–56.

[61] R. Norris, D. Carroll, and R. Cochrane, "The effects of physical activity and exercise training on psychological stress and well-being in an adolescent population," *Journal of psychosomatic research*, vol. 36, no. 1, pp. 55–65, 1992.

[62] "Oxycon Mobile," http://www.carefusion.com.

[63] J. Pan and W. J. Tompkins, "A real-time qrs detection algorithm," *Biomedical Engineering, IEEE Transactions on*, no. 3, pp. 230–236, 1985.

[64] D. J. Paterson, G. A. Wood, A. R. Morton, and J. D. Henstridge, "The entrainment of ventilation frequency to exercise rhythm," *European journal of applied physiology and occupational physiology*, vol. 55, no. 5, pp. 530–537, 1986.

[65] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[66] R. M. Rangayyan, *Biomedical signal analysis*. IEEE press New York, 2002.

[67] M. Rofouei, M. Sinclair, R. Bittner, T. Blank, N. Saw, G. DeJean, and J. Heffron, "A non-invasive wearable neck-cuff system for real-time sleep monitoring," in *BSN*, 2011.

[68] A. Sadeh, K. Sharkey, and M. Carskadon, "Activity-based sleep-wake identification: An empirical test of methodological issues," in *Sleep*, vol. 17, no. 3, 1994, pp. 201–207.

[69] P. Soille, *Morphological image analysis: principles and applications*. Springer-Verlag New York, Inc., 2003.

[70] B. C. Sporer, G. E. Foster, A. W. Sheel, and D. C. McKenzie, "Entrainment of breathing in cyclists and non-cyclists during arm and leg exercise," *Respiratory physiology & neurobiology*, vol. 155, no. 1, pp. 64–70, 2007.

[71] M. Sumida, T. Mizumoto, and K. Yasumoto, "Estimating heart rate variation during walking with smartphone," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013, pp. 245–254.

[72] S. Villard, J.-F. Casties, and D. Mottet, "Dynamic stability of locomotor respiratory coupling during cycling in humans," *Neuroscience letters*, vol. 383, no. 3, pp. 333–338, 2005.

[73] "Voyager Legend Mobile Bluetooth Headset," http://www.plantronics.com/us/product/voyager-legend.

[74] "Voyager Pro HD Mobile Bluetooth Headset," http://www.plantronics.com/us/product/voyager-pro-hd.

[75] J. Webster *et al.*, "An activity-based sleep monitor system for ambulatory use," *Sleep*, vol. 5, no. 4, pp. 389–399, 1982.

[76] A. Zhan, M. Chang, Y. Chen, and A. Terzis, "Accurate caloric expenditure of bicyclists using cellphones," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM, 2012, pp. 71–84.