NORTHWESTERN
UNIVERSITY

Electrical Engineering and Computer Science Department

Technical Report
NWU-EECS-11-09
August 31, 2011

Acoustic Sensing of Location and User Presence on Mobile Computers

Stephen P. Tarzia

Abstract

This dissertation evaluates the claim that "acoustic sensing on mobile computers can provide accurate user presence and location information while requiring no new hardware." We introduce two new acoustic sensing techniques for computers; one is a sonar system for detecting user presence, the other is an ambient sound fingerprint to support determining location while indoors. The common goal is to sense the physical context of the computer; this problem at the intersection of mobile, ubiquitous, and pervasive computing systems research.

Our sonar presence detection system senses motion in the environment by emitting an inaudible ultrasonic tone from the computer's speaker while using the microphone to listen for variations in the tone's echo. A user study showed that motion can be reliably sensed even in cases when the user is simply watching the display and thus no movement is expected. We applied sonar presence detection to the problem of display power management in a publicly-released software utility.

This work was made possible by support from a Dr. John N. Nicholson fellowship and the National Science Foundation (NSF) via grants CNS-0720691 and CNS-0347941.

We next introduce a way for portable computers such as smartphones to determine their location while indoors by listening to ambient sounds. No new hardware is required on the device and no infrastructure is required in the environment. Sensing is entirely passive, using the device's microphone. Surprisingly, we found that each location has a relatively unique background sound. In other words, the "quiet" sound of each room is distinct and we introduce some filtering techniques which can measure the background sound even when there is noise, such as a person talking. Our technique works both in closed rooms and open hallways and even in cases when the user carrying the device is walking.

This dissertation uses an empirical approach, measuring real-world performance of our systems with user studies when possible. As part of these experiments, we have publicly released useful and compelling research software. This software, and our experimental results, show that background sounds and ultrasound have much more practical use than we might expect.

Keywords: localization, sonar, presence, audio, sound, signal processing, power management, ultrasound

NORTHWESTERN UNIVERSITY

Acoustic Sensing of Location and User Presence on Mobile Computers

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Electrical Engineering and Computer Science

By

Stephen P. Tarzia

EVANSTON, ILLINOIS

August 2011

Abstract

This dissertation evaluates the claim that "acoustic sensing on mobile computers can provide accurate user presence and location information while requiring no new hardware." We introduce two new acoustic sensing techniques for computers; one is a sonar system for detecting user presence, the other is an ambient sound fingerprint to support determining location while indoors. The common goal is to sense the physical context of the computer; this problem at the intersection of mobile, ubiquitous, and pervasive computing systems research.

Our sonar presence detection system senses motion in the environment by emitting an inaudible ultrasonic tone from the computer's speaker while using the microphone to listen for variations in the tone's echo. A user study showed that motion can be reliably sensed even in cases when the user is simply watching the display and thus no movement is expected. We applied sonar presence detection to the problem of display power management in a publicly-released software utility.

We next introduce a way for portable computers such as smartphones to determine their location while indoors by listening to ambient sounds. No new hardware is required on the device and no infrastructure is required in the environment. Sensing is entirely passive, using the device's microphone. Surprisingly, we found that each location has a relatively unique background sound. In other words, the "quiet" sound of each room is distinct and we introduce some filtering techniques that can measure the background sound even when there is noise, such as a person talking. Our technique works both in closed rooms and open hallways and even in cases when the user carrying the device is walking.

This dissertation uses an empirical approach, measuring real-world performance of our systems with user studies when possible. As part of these experiments, we have publicly released useful and compelling research software. This software, and our experimental results, show that background sounds and ultrasound have much more practical use than we might expect.

Thesis committee

Peter A. Dinda (chair), Northwestern University
Gokhan Memik, Northwestern University
Bryan Pardo, Northwestern University
Robert P. Dick, University of Michigan

Acknowledgments

I would like to thank my thesis committee Peter A. Dinda (chair), Gokhan Memik, Bryan Pardo, and Robert P. Dick for their judicious guidance.

Most of the text of Chapters 2, 3 and 4 was taken directly from publications co-authored by Professors Peter A. Dinda, Robert P. Dick and Gokhan Memik [TDDM09, TDDM10, TDDM11]. These collaborators deserve credit for writing some passages as well as editing all of that text.

This work was supported by a Dr. John N. Nicholson fellowship and by the National Science Foundation under awards CNS-0720691 and CNS-0347941.

Contents

List of Figures

Chapter 1

Introduction

This dissertation addresses the broad question of "what new information can be extracted from sound on mobile computers?" It turns out that background sounds and ultrasound have much more practical use than we might expect. Before describing our specific findings, I will give some motivation for studying sound.

1.1 Sound and acoustic sensing

One appealing feature of acoustic sensing is that sound is a very easy to capture. Miniature electret condenser and MEMS (silicon) microphones are very cheap and they require very little electricity to operate. Thus, we find them on countless electronic devices. Many current smartphones have multiple microphones: one at the bottom of the device for telephone conversations, another at the back of the device for video capture and noise cancellation, and perhaps a third wireless Bluetooth microphone clipped to the user's ear. These devices also have speakers which can be used to acoustically probe the environment. Sound data is not just readily available but also relatively easy to process. It is one-dimensional and typical sound processing tasks require little time, energy and memory. So, there is tremendous potential for ubiquitous "hearing-enabled" software. This dissertation aims to develop theory and techniques to promote new classes of sound-aware software applications.

This dissertation develops acoustic means of sensing *physical context*. The goal here is to determine what is happening physically around the computer. Such sensing is critical for two related fields: *mobile computing* and *ubiquitous and pervasive computing*.

1.2 Context-aware systems

Imagine that you are a technology-savvy person living in a modern city. If you have a large discretionary spending budget, you probably own and carry a sophisticated "smartphone" which allows you to access the Internet, email, and your social network anywhere the device can get a cellular radio signal. If you need a cup of coffee you can quickly and easily find a cafe nearby. By simply typing "cafe" in your smartphone's web search application you instantly get a list of nearby cafes with reviews, hours of operation, and directions.

This scenario is made possible by the phone's ability to determine its location using GPS, Wi-Fi and cellular radio signals. Without automatic location sensing, you would have to enter not just the word "cafe" but also include a precise description of the location, such as "2145 Sheridan Rd, Evanston" or "60208" (the postal/zip code) in order to see relevant results. This requirement for additional input is a real inconvenience because smartphone keyboards are small and error-prone. Also, you may be in an unfamiliar location and even if you know where you are, it is very likely that you would not be able to recall a precise machine-readable description of that location. Instead, you may describe the location using ambiguous natural language which a computer would have trouble interpreting. For example, you may describe the location as "past the Mexican restaurant near Alice's house." Even if these directions were given to a human, several clarifying questions might be needed to pinpoint the location.

Software that automatically takes into account transient information about the physical environment or the user is said to be *context-aware*. Location sensing is one example but there are many others and generally they rely on *sensors* that monitor the surroundings to automatically infer the current context. For example, during a call, the phone's proximity sensor allows it to determine whether you are holding the phone up to your ear or holding it out in front of you. In the first case, the touchscreen must be disabled to prevent unwanted button clicks and the display can be powered-off to lengthen battery life. In the latter case the speakerphone can be automatically enabled. The phone's accelerometer can be used to determine its orientation and to flip the screen contents automatically. These are all examples of sensing phone context to improve the user experience.

Sensors have already made smartphones much more useful and engaging than the previous generations of portable electronics. However, we have only scratched the surface of what is possible with on-device sensors. Let us imagine for a moment the futurist promise of portable or pervasive computers which are endowed with sophisticated artificial intelligence and which serve as virtual secretaries, butlers, fitness trainers, and even friends. These hypothetical computers would need very detailed knowledge of the physical world surrounding them. This is part of the *pervasive and ubiquitous computing* vision; a vision which considers computers not

as appliances with display panels and buttons but rather as computing services and agents which surround us in *our world*. In order for computers to inhabit our world they need very sophisticated sensing abilities. In this dissertation I will focus on the short-term applications of two new sensing abilities. These are not the end-goals but steps along the path to this long-term vision.

This dissertation considers two acoustic sensing abilities which are useful in scenarios where the user is roaming about the physical world while using computing services. User *movement* has become a critical feature of popular computing systems and it is central to this dissertation. Modern computer users increasingly access services "on the go" due to device miniaturization and improving wireless networks. User location has become dynamic and we have seen that many computing services can perform better by being location-aware.

Presence and engagement First, let us assume that the computers are installed throughout the environment. This could mean desktop and laptop computers at desks in an office building or it could mean a high density of sensor nodes and displays as in the pervasive computing vision. In either case, each computer should sense the presence of users and their level of engagement in order to determine when services should be offered versus when energy can be saved by powering-down parts of the system.

Localization Alternatively, we may assume that portable computers, such as smartphones, are being carried by the users as they roam about. If the device knows its current location, then it can tailor the services it offers to the user to match that location. The process of determining the location of a mobile computer is called *localization*. Outdoors, localization is provided by GPS. However, indoor localization is quite challenging and it is an open research problem.

1.2.1 Presence- and engagement-based services

The following services are enabled by presence sensing.

Power management. As we surround ourselves with more and more computers, leaving them constantly running becomes very wasteful. At the same time, manually switching every device on and off is onerous. Presence sensing allows devices to remain on only when they might be needed by a nearby person. Saving energy in this way is especially important for battery-powered devices.

Disappearing computers. Glowing and blinking computers around us are not just energy-wasteful they are also distracting. "Calm computing" [WB97] and "disappearing computers" [RSW05] are visions

that depend on both presence sensing and intention inference to make computers that offer services when appropriate, and get out of the way otherwise.

Automatic suspension. There are also cases when we would want a computer application to automatically and quickly respond to the user's departure. The operating system might lock the screen to prevent unauthorized access, a game might pause its action, or the system might log-out or restart to prepare for the next user.

1.2.2 Location-based services

The following services are enabled by accurate indoor localization.

Environment interaction. Localization allows users to control and receive feedback from devices in their environment. Instead of installing a display and button panel on every device, the user may use their handheld computer as a universal remote control; localization would allow a relevant list of controls to be displayed. For example, upon entering a conference room the user's smartphone can display buttons for controlling the projector screen, blinds, and the climate control temperature for that room. Users can also interact with non-electronic features such as "clicking" a poster to sign up for an event [ACH+01] or make a purchase.

Reminders. A big problem with electronic to-do lists is that they can become so long that they are ignored. A solution to this problem is to highlight reminders that are relevant to the current location. For example, telling the user to take an umbrella when they are about to leave their home [DA00, LFR+06]. Co-location can also be used to trigger reminders; for example, remind me to discuss some topic when I encounter a certain colleague [DA00]. Calendar software can trigger a reminder if (and only if) the user is not at a scheduled meeting's location.

Targeted advertising. vendors may want to promote certain items when the user is located in a particular store [ACC09].

Tour guides. A few researchers have developed interactive tour guide devices for museum visitors [AAH+97, CP04]. Such devices dynamically show visitors information about the nearby exhibits.

Navigation aids. If the user is in an unfamiliar place they may simply want to see where they are on a map; they might also want walking directions to reach some destination. Indoor navigation aids would be useful in large buildings such as shopping malls, conference centers, hospitals, etc.

1.3 Thesis statement

At the core of this dissertation are two new sensing modes for context-aware mobile and ubiquitous computing that aim at improving the user experience. I claim the following.

Thesis statement Acoustic sensing on mobile computers can provide accurate user presence and location information while requiring no new hardware.

User presence and location are both examples of physical context which are extraordinarily powerful, as illustrated in the examples above. Essentially, we want to answer the following two questions using sound.

Is someone there? In Chapters 2 and 3 we introduce an ultrasonic sonar system that uses the computer's speaker and microphone to probe the surroundings. It detects the presence of a person by the tiny movements than even still people make. It can reliable distinguish between a person sitting in front of the computer watching a video and an empty chair.

Where are we? In Chapter 4 we introduce a way of extracting a unique location "fingerprint" from a recording of the location's ambient sound. We use this fingerprint to accurately determine the current location using sound alone. The dozens of quiet rooms we sampled actually do sound different and their sound remains the same when revisited months later.

My thesis statement is well-supported by these two results; the accuracy that these systems achieve is indeed surprising.

1.4 Related work

Ours is certainly not the first mobile acoustic sensing work, and I give a detailed account of relevant past work in Sections 2.1.1, 3.2, 4.2, and 5.1.2. Still, our work has pushed what was known to be possible with acoustic sensing. We are the first to determine location and presence with sound alone. Here, I will highlight two important examples of past work to show how our work stands out.

FaceOff, by Dalton and Ellis [DE03], is a software technique for detecting whether a human is sitting in front of a laptop computer. It does this by processing images captured by a webcam. We introduce a technique, described in Chapter 2, that instead uses sonar for the same task. The advantages of our technique are its simplicity, reliability, and low computational burden. This is an example showing the advantage of acoustic sensing relative to other modes of sensing.

SurroundSense, by Azizyan et al. [ACC09], is a software technique for mobile phones that uses the radio, motion sensors, camera, and microphone to determine in which of several neighboring shops the phone is currently located. They characterize each shop with a distribution of observed loudness levels and use this acoustic evidence to rule-out location hypotheses with very different loudness distributions. SurroundSense was the first work to apply acoustic sensing to localization, but its results are not surprising. Our own acoustic sensing technique, described in Chapter 4, shows that much more is possible with the microphone; we achieve accurate indoor localization using the microphone alone. Surprisingly, we are able to distinguish dozens of rooms that have the same loudness characteristic – they are all quiet.

1.5 Methods

In this dissertation I focus on *software-based* techniques that use audio hardware that is already built-into the target devices. This approach has several advantages. It reduced our own prototype-building time, since software is generally easier to build and modify than hardware. More importantly, it allowed us to distribute our prototype systems via the Internet to other researchers and to thousands of users.

We use an experimental research method that emphasizes evaluating the *real-world performance* of our systems. There are three steps.

1. Build a fully-functional software prototype for the system.
2. Release the software binaries and source code to the public.
3. Analyze performance of the system using data provided by test users and from our own tests.

When possible, we gather data from real human usage by running *user studies*. The experimental method described above is common in computer systems research and it leads to *empirical results* rather than extensive theory.

1.6 Contributions

Our research contributions can be summarized as follows.

- To human-computer interaction (HCI) research, we contribute a new method of sensing the presence of users at computers using ultrasonic sonar [TDDM09], as described in Chapter 2. This technique is reliable, has low computational overhead, and requires no new hardware. It is also the first *active* acoustic sensing system we have seen that uses standard computer audio hardware.

- To computer systems research, we contribute a valuable study of display power management policies, including a huge computer usage data set [TDDM10]. We also give additional evidence of the benefit to system evaluation of adding user measurement. This is described in Chapter 3.

- To mobile and ubiquitous computing research, we contribute a sound-based indoor localization technique which works in the absence of any infrastructure or specialized hardware [TDDM11], as described in Chapter 4.

- We also introduce a new instance-based Markov localization technique and quantify the benefit of using a history of sensor readings for indoor localization, as described in Chapter 5. This work also shows that acoustic localization can work while walking and in hallways.

- Throughout all this work I have released open-source software implementing our research tools for the benefit of end-users and developers. This includes

  - *Sonar Power Manager* for Windows and Linux, which has been downloaded over 13,000 times, and
  - *Batphone* for iOS (iPhone, iPod, and iPad), which has been downloaded over 1,000 times,

  as well as data sets and scripts for replicating our published results.

Chapters 2 through 5 describe experiments which were conducted in the order presented. All are related to acoustic sensing, although Chapter 3 also devotes some time to display power management, an application for sensing. These chapters are somewhat self-contained, each presenting its own introduction and conclusions. I conclude the dissertation in Chapter 6 by discussing how my results support the thesis statement and by discussing future work.

Chapter 2

Presence

This chapter describes a technique for detecting the presence of computer users which is useful for the ubiquitous computing applications described in the previous chapter. We use sonar to sense the user; as we might expect, human bodies have a different effect on sound waves than air and other objects. What is surprising is that we are able to build an ultrasonic sonar sensing system using standard microphones and speakers, such as those built-into laptop computers. This is illustrated in Figure 2.1. Thus, ours is *software-based* sonar. We conducted a user study in which 20 volunteers used a computer equipped with our ultrasonic sonar software. Our results show that it is possible to detect the presence or absence of users with near perfect accuracy after only ten seconds of measurement. We find that this technique can differentiate varied user positions and actions, opening the possibility of future use in estimating attention level.

2.1 Introduction

As mentioned in Section 1.2.1, there are many cases in which user presence sensing would be useful. In ubiquitous computing systems, it is often advantageous for distributed electronic devices to sense the presence of roaming humans, even when they are not directly interacting with the devices. This ability allows such a system to provide services to users only when appropriate. In traditional desktop computing, attention information is also useful; it is already used by Operating System (OS) power management systems to save energy by deactivating the display when the keyboard and mouse are inactive. Security systems prevent unauthorized access by logging out or locking a user's session after a timeout period. In both of these cases, the OS must know whether a user is present and *attentive*, i.e., using the computer system, or absent.

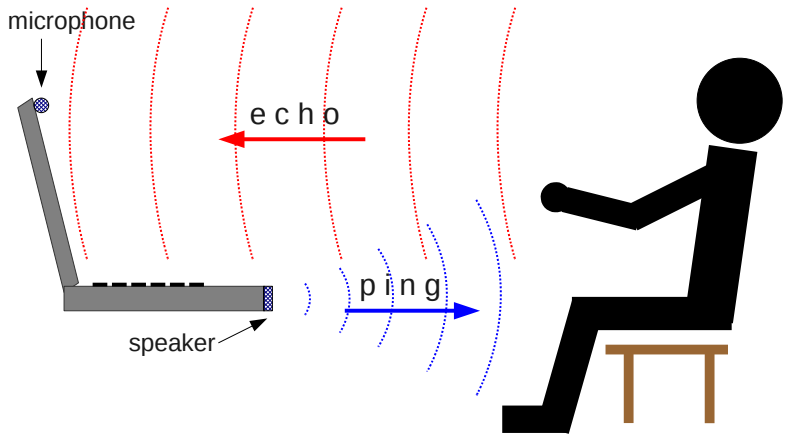


Figure 2.1: Sonar presence sensing schematic.

User attention state	Definition	Activity	User-study task
Active	Manipulating the keyboard, mouse, etc	*Typing*	Replicating an on-screen document on a laptop using a word processor
Passively engaged	Reading the computer screen	*Video*	Watching a video on the laptop's display
Disengaged	Sitting in front of the computer, but not facing it	*Phone*	Short multiple-choice telephone survey using telephone next to the laptop
Distant	Moved away from the computer, but is still in the room	*Puzzle*	Pencil-and-paper word-search puzzle on the desk beside the laptop
Absent	User has left the room	*Absent*	After the participant left the room

Figure 2.2: Proposed user attention states and each state's associated user-study task.

We have identified five different human user *attention states* among which an *ideal* system would distinguish, shown in Figure 2.2. These states were chosen to represent cases commonly encountered in the personal desktop computing scenario. The active state is trivially detectable using input activity. Our ultimate goal is to distinguish the remaining four attention states. In this chapter, however, our evaluation focuses on specific *activities*, as shown in the table. A given attention state may be as associated with numerous activities.

2.1.1 Related work

Sonar has been well-studied for a variety of applications. For example, mobile robots are often equipped with sonar range-finders, such as those manufactured by Polaroid. These sonar sensors measure the time delay of echos to determine the distance of objects from the sensor. Some such systems use arrays of sensors for beam-forming; this allows the sensing to be done in a precise direction. These traditional sonar systems are quite different that what we propose here since we are doing motion detection, not ranging.

Motion sensing with sonar is not a new idea. Sonar-based motion sensors are commonly used in multimedia displays and in building lighting systems. Our work is different in that it is entirely software-based, using hardware which was not designed for ultrasound or sonar. We have shown how sonar motion sensing can be added to any device having a microphone and speaker.

In the ubiquitous computing community, this work falls under the category of *Activity detection*. Ubiquitous computing research systems have attempted to determine a human user's current activity using data from a variety of sensors such as GPS, RFID, infrared motion detectors, accelerometers, and video cameras. Our work complements past work but differs in that it uses hardware that is already available in many home and office electronic devices.

In the OS community, we know of only one prior research project that studies user attention detection: "FaceOff" tackles the fine-grained OS power management problem [DE03]. It processes images captured by a webcam to detect whether a human is sitting in front of the computer. After our work's publication, Kim et al. [KKK+11] applied gaze detection to distinguish between attentive and inattentive states.

Ultrasonic sounds have such high frequency that humans cannot hear them. They have already been used in context-aware computing for several different tasks. Madhavapeddy et al. used ultrasonic and audible sound as a short-range low-bandwidth wireless communication medium [MSS03, MSST05]. The Cricket localization system by Priyantha et al. uses ultrasonic and radio beacons to allow mobile devices to determine their location within a building [PCB00]. Borriello et al. built another room-level location service

similar to Cricket [BLO+05]. Peng et al. built a ranging system for pairs of mobile devices that uses audio [PSZ+07]. In this work we propose using ultrasonic sound for another task: directly sensing the user.

2.1.2 Active sonar

Sonar systems emit sound "pings" and sense the resulting echoes. Based on the characteristics of the echoes, a rough map of the surrounding physical space can be derived. Sonar is used by animals, such as bats and dolphins, for navigation and hunting [TMV04]. Man-made systems have been invented for fishermen, divers, submarine crews, and robotics. The omnidirectional (unfocused) and relatively insensitive microphones and speakers built into most laptops are not ideal for building a precise sonar system. However, our expectations for the sonar system are modest; we only need information about the user's activity, not a detailed map of the room.

Audio in the 15 to 20 kHz range can be produced and recorded by a laptop computer but is inaudible to most adults [Moo03]. Thus, by using these audio frequencies and assuming the absence of children and pets that are sensitive to ultrasound, we can program a sonar system that is silent to the user. Our sonar system emits a continuous high frequency (ultrasonic) sine wave and records the resulting echoes using a microphone.

2.2 Hypotheses

What characteristics of the echoes might vary with user activity? We make the following two conjectures:

1. The user is a close surface that will reflect sound waves emitted from the speaker.
2. The user's presence may affect the amount of reflection and therefore the *intensity* of echoes received by the microphone.

In many scenarios the user is the only moving object near the computer. It might therefore be helpful to listen for signs of movement in echoes; any data related to movement is likely to be related to the physically-active user's behavior. In particular, motion in the environment is likely to introduce additional variance in the echoes since the angles and positions of reflection surfaces will be changing. Thus, the user's presence and activity should affect the *variance* of echo intensity. Our results, presented later, support this claim.

2.3 User study

We conducted a user study to determine how sound echoes vary with changes in user attention state. We were specifically interested in how echo intensities and variances are affected. Our study protocol was reviewed and approved by our university's Institutional Review Board and is described briefly in this section. We recruited twenty paid volunteers from among the graduate students in our department. During the study, participants spent four minutes working on each of four tasks. Each task, plus absence, shown in Figure 2.2 is associated with one of five attention states. The participants were not told any details about the sonar system and in particular they were unaware that their motion level was of interest to us.

While the users completed the tasks, a 20 kHz sine wave was played, and recordings of the echoes were made. We chose a constant 20 kHz sine stimulus because such a signal is likely to be both silent to the user and sensed by the hardware. Any time-varying signal signal would have lower-frequency components which the user would hear; for example, switching the signal off or on results in a clicking sound. We chose a narrow-band signal because we expect very little bandwidth to be available between the top of the user's hearing frequency range and the top of the hardware's frequency response. Also, the simplicity of a sine stimulus led to a straightforward feature extraction stage. The disadvantage of a sine stimulus is that we cannot compute the echo delay; thus ranging is not possible.

A secondary goal of the study was to determine which types of speakers and microphones would be suitable for a computer sonar system. We therefore experimented with combinations of four microphones and four speakers, leading to sixteen recordings being made for each task. As illustrated in Figure 2.4, the four microphones recorded simultaneously. The four minutes that each participant spent on a task was divided into four one-minute intervals. During each interval a different speaker played the sine wave. In this way, a recording for each combination of microphone and speaker was obtained for each user performing each task. To eliminate temporal biases, the order of tasks completed and speaker activations within those tasks were randomized for each user (except that the "absent" task always occurred last, after the user had left). The total user study duration was twenty minutes: four minutes for each of five tasks.

2.3.1 Experimental setup

Our experimental setup was modeled after an office environment. The audio equipment and laptop computer were arranged on a large desk as shown in Figure 2.5 and the participant sat in a rolling office chair. The study administrator was seated at an adjacent desk throughout the study. Everything, including the word puzzle clipboard was fastened securely to the desk to ensure consistency between runs. The telephone cord

Microphones	*internal:*	Laptop's internal microphone, located near the touchpad
	ST55:	Sterling Audio ST55 large diaphragm FET condenser mic connected through Edirol UA25 USB sound card
	PC:	Inexpensive generic PC microphone connected via Plantronics USB DSP v4 sound card
	webcam:	Built-in microphone on a Logitech Quickcam 3000 pro USB webcam
Speakers	*internal:*	The laptop's internal speakers, located on either side of the keyboard.
	sound-sticks:	Harman Kardon SoundSticks USB speakers that include a subwoofer, left, and right speakers.
	dell:	Dell's standard desktop computer speakers connected via Plantronics USB DSP v4 sound card
	null:	Record without any emitted sound wave

Figure 2.3: Audio hardware used in user study.

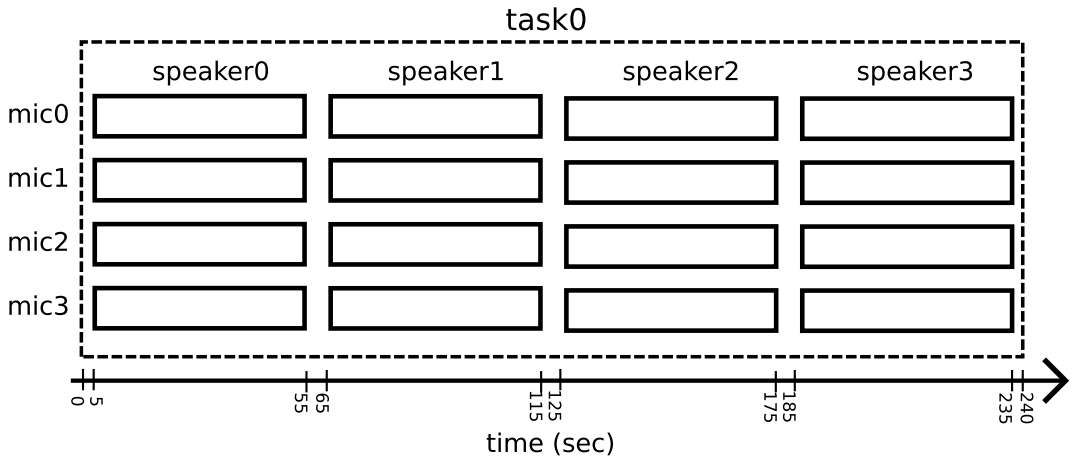


Figure 2.4: Timeline of all sixteen recordings taken while a user study participant completes a single task.

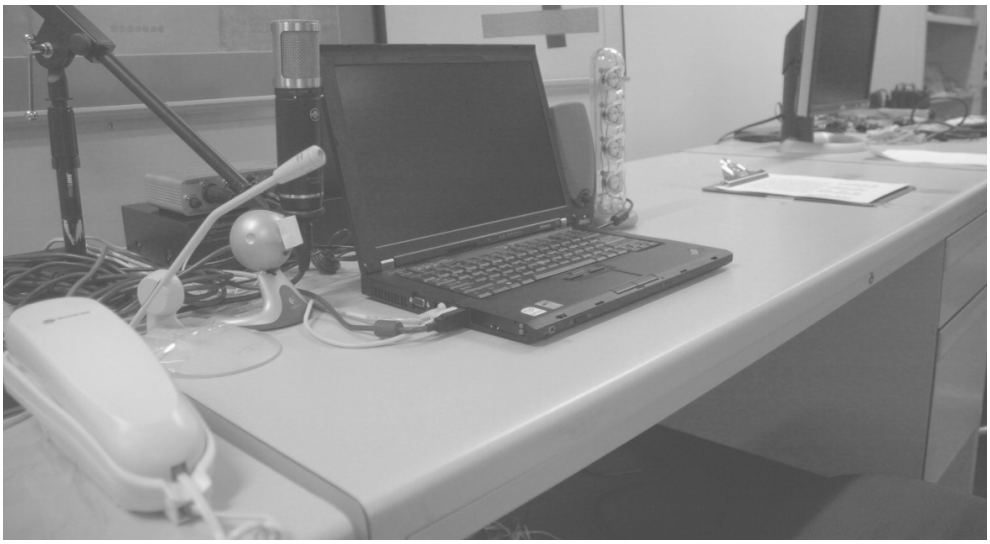


Figure 2.5: User study setup.

was shortened to force users to remain in front of the laptop while making calls. A Lenovo T61 laptop with a 2.2 GHz Intel T7500 processor and 2 GB RAM was used. The OS was Ubuntu Linux.

Setup details are as follows. We used the audio hardware listed in Figure 2.3 which represents a wide range of microphone and speaker types. The speaker volumes were set to normal listening levels. We used the right-hand side speakers only, for simplicity. We chose a sonar frequency of 20 kHz because very few people can hear tones at this frequency. Recording and playback audio format was signed 16 bit PCM at 96 kHz sample rate (almost all new laptops support these settings). The first and last five seconds of each recording were discarded leaving a set of fifty-second recordings for analysis.

2.3.2 Feature extraction

Analysis of the recordings was done after the user study was complete. We wrote Python scripts to analyze the 18 GB of WAV files using standard digital audio signal processing techniques. In this section we describe how echo intensities were calculated from the recordings and we describe a feature of these intensities, called *echo delta*, which we used when explaining our results.

To calculate an estimate of the echo intensity, we use a frequency-band filtering approach. We assume that all of the sound energy recorded in the 20 kHz band represents sonar echos; our measurements confirm that ambient noise in that frequency-band was negligible. We use Bartlett's method (with 10 non-overlapping rectangular windows and a 1024-point Fast Fourier Transform (FFT)) to estimate the recording's power spectrum [PM96]; in each of the ten windows, the amplitude of the Fourier coefficient nearest 20 kHz was squared to get an energy value and then averaged with the other nine values. As is common in audio measurement, we scaled down the results with a base-10 logarithm to get decibel units.

In our results, we use a characterization of the echo's variance that we call *echo delta*. To calculate the echo delta of each recording we first break it into a sequence of 100 ms windows. The echo intensity is calculated for each of these by Bartlett's method, as described above; this gives us a sequence of echo intensity values $e_1 ... e_N$. The echo delta Δ_e is then just the average of absolute differences in that sequence:

$$\Delta_e(e_1 ... e_N) \equiv \frac{1}{N} \sum_{i=1}^{N-1} |e_{i+1} - e_i|$$

Echo delta characterizes echo variances on the time scale of a single echo intensity window, i.e. 100 ms.

In the next chapter, we describe a Windows and Linux software utility which calculates echo delta in real time (see Section 3.5). The source code for that software has been published on the web, and that

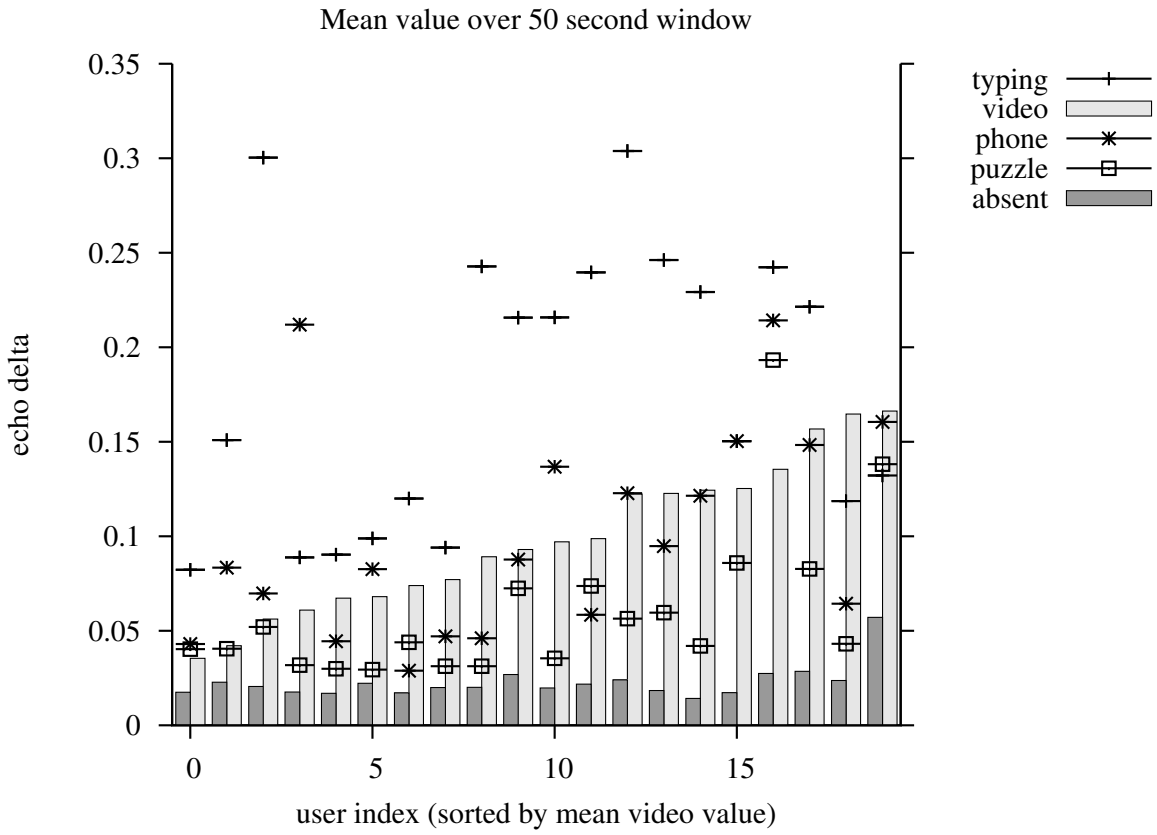


Figure 2.6: User study echo delta (Δ_e) sonar measurements for soundsticks–webcam hardware combination taken over the full 50 second observation. Note the clear difference between measurements of users during video and phone activities versus the absent state.

C++ code serves as a reference implementation for echo delta presence detection. As an optimization, that implementation uses Goertzel's algorithm to compute a single Fourier coefficient at 20 kHz rather than using an FFT to compute 1024 coefficients.

Before settling on the above feature extraction steps, we tried using several alternative statistics of the echo intensity sequence to to capture motion. In particular, we considered the variance of the echo intensity sequence. However, this gave inferior presence detection results. Somehow, echo delta's insensitivity to long-term changes in echo intensity seems to be beneficial.

2.4 Results

We now quantify the effect of user state on sonar measurements in our user study. Although our experiments included three different speakers and four microphones, for brevity, we fully present results from only one

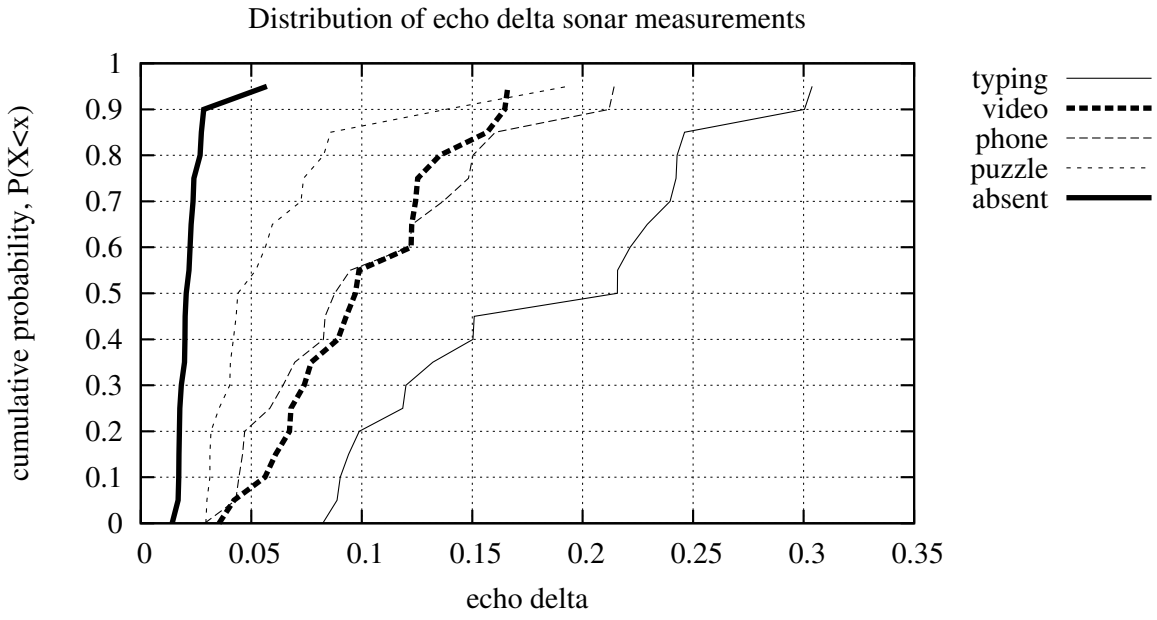


Figure 2.7: Cumulative distribution of mean echo delta values plotted in Figure 2.6.

combination: those obtained using the soundsticks speaker and webcam microphone.

A comparison of echo delta (Δ_e) among different activities is compelling. Figure 2.6 shows Δ_e for each study participant, in each of the five attention states. There is a clear trend of increasing Δ_e when moving from absent to more engaged user states. The exact ordering of the middle states (video and phone in particular) varies between users, but all for all users we observe an increase in Δ_e with their presence in any of the four attention states. Figure 2.7 shows a distribution view of the same data.

To test the potential responsiveness of our sonar system, we simulated a reduction in the recording time window by splitting each fifty-second recording into five 10-second windows. Figure 2.8 shows the range of Δ_e values calculated in these smaller windows for a representative pair of states. We can see that, as compared to Figure 2.6, the gap between the video and absent states is narrowed, but the two still do not intersect. This demonstrates a trade-off between time window size and state identification accuracy.

In Figures 2.6 and 2.8, there is a clear difference between users who are absent and those who are present but not interacting directly with the machine. Combined with traditional Human Input Device (HID) monitoring, the proposed sonar approach makes it possible to differentiate between interactive users, present but non-interactive users, and absent users.

Similar, but weaker, results were obtained from several other microphone and speaker combinations, as presented later in Figure 2.11. Figure 2.9 summarizes the sonar sensitivity of each hardware combination.

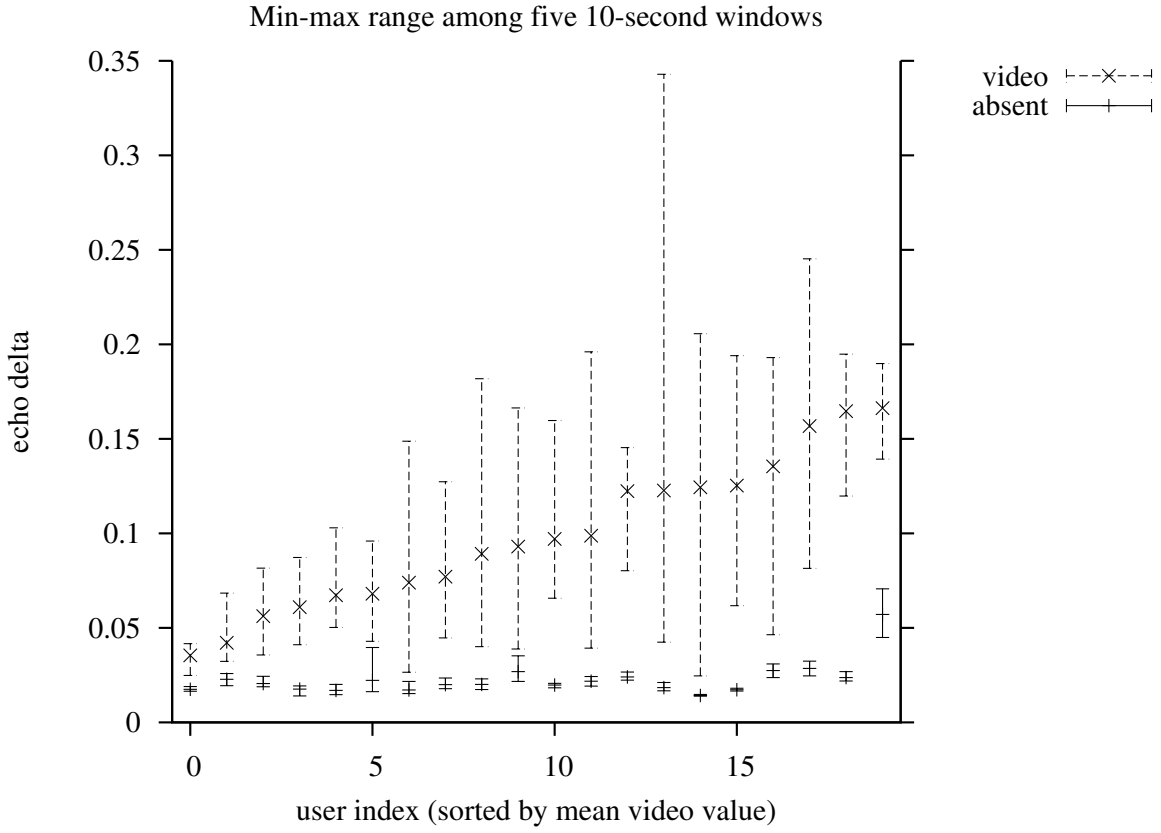


Figure 2.8: User study echo delta (Δ_e) sonar measurement range for soundsticks–webcam hardware combination for five 10-second windows within the 50 second observation. Note that the distinction between video and absent states seen in Figure 2.6 remains.

Speaker	*Microphone*			
	internal	ST55	PC	webcam
internal	3.84	19.3	0.343	6.49
soundsticks	20.5	29.6	11.8	*21.3*
dell	7.06	25.6	13.8	16.6
null	0	0	0	0

Figure 2.9: Ultrasound sensitivity for various combinations of audio hardware, measured in decibel (dB) signal to noise ratio (SNR). Speaker and microphone combinations with higher SNRs are likely capable of better sonar performance, as confirmed later in Figure 2.11.

This is measured with the decibel signal to noise ratio (SNR) of the sine wave's echo; it is the ratio of the recorded 20 kHz energy with the speaker on versus off. The video activity was used for these calculations. Ultrasound sensitivity varies with the hardware choice but does not seem to depend on hardware cost. It is particularly noteworthy that the webcam's relatively basic microphone was a top performer. We suggest that microphone and speaker positioning and omission of high-frequency noise-filtering circuitry are the most important factors for good sonar performance. It is important to note that the SNR depends on the relative positions of the microphones and speakers, as shown in Figure 2.5. Depending on the positions, varying amount of direct and reflected sound will be captured by the microphone. Thus, the SNRs reported in Figure 2.9 are only useful for approximations.

Figure 2.9 shows that, in the user study, the Lenovo T61 laptop's built-in audio hardware performed poorly. However, sonar performance varies between laptop models. After developing the Sonar Power Manager software described in Section 3.5, we found that many other models of laptop computer performed well using their built-in audio hardware. We suggest that laptop manufacturers could choose ultrasound-capable audio hardware without incurring any additional costs. The details of this laptop population study appear in Section 3.7.6.

Processing overhead for sonar is negligible. As an indication, the analysis runtime for a fifty-second sample was only 1.6 s on our study laptop. Therefore, a real-time-processing implementation would add a load of about 3% to one of the CPU cores. Furthermore, the energy overhead of activating the audio hardware is negligible compared to display or CPU energy.

2.5 State classifier

Encouraged by the results shown in Figure 2.8, we built a binary state classifier to automatically distinguish between passively engaged (video) and absent states. We use a very simple threshold-based classification scheme. Analyzing the training data gives an average echo delta Δ_e for the two states: $\Delta_e^{passive}$ and Δ_e^{absent}. We choose a threshold K between the two values using a weighted geometric mean: $K \equiv (\Delta_e^{passive} * (\Delta_e^{absent})^2)^{1/3}$. To classify the test data, we simply compare its Δ_e to K. If it is greater than or equal to K we classify it as passively engaged, otherwise absent. Section 3.4.2 presents a method for choosing K in cases when explicit state training is not possible or is too onerous.

Figure 2.10 shows a confusion matrix for the binary state classifier on the user study data. The fifty second recordings from the passively engaged and absent states were broken into ten second windows as in Figure 2.8. For each user, one passively engaged window and one absent window were used as training.

Actual state	*Predicted state*	
	passively engaged	absent
passively engaged	0.957	0.043
absent	0.031	0.969

Figure 2.10: Confusion matrix for binary presence classifier using 10 s of training and 40 s of test recordings.

Speaker	*Microphone*			
	internal	ST55	PC	webcam
internal	59.2%	84.6%	47.4%	88.4%
soundsticks	85.0%	79.8%	59.8%	*96.3%*
dell	88.0%	93.0%	68.5%	74.6%
null	50.5%	54.9%	43.5%	72.7%

Figure 2.11: Presence detection accuracy for various combinations of audio hardware, averaged across all users. Speaker and microphone combinations with higher SNRs in Figure 2.9 generally perform better. The "null" speaker is included to give an indication of error margins.

The remaining four windows from each state served as test data. Classification was repeated using every pair of absent and passively engaged windows as training. False positive and false negative rates were both below 5%. After increasing the length of the training data window to 25 s, classification of the remaining 25 s window became perfect.

We can define the overall presence detection accuracy as the mean of the true positive and true negative rates for binary classification. Figure 2.11 shows how presence detection accuracy varies across the audio hardware that we tested. This generally correlated well with the SNRs shown in Figure 2.9. One exception is the null-speaker/webcam-microphone combination. We assume that the relatively good performance here is due to the introducing some ultrasound through their own actions.

Note that distinguishing between the four attentive states is much more difficult than the above binary classification. This is evident in the reordering of states among different uses seen in Figure 2.6. For example, it is not clear that distinction between video and phone activities is possible, but this was expected since users' behaviors and postures for these activities are varied. Nonetheless, by also monitoring HID events, we can clearly distinguish between three states: active, passively engaged, and absent.

2.6 Conclusions

The experimental results support the hypothesis that the user's presence indeed causes changes in echo intensity. More generally, we have demonstrated that sonar implemented using commodity computer hardware

can measure useful information with low computational burden. Our user study was performed on a laptop computer and used traditional desktop computing applications. However, any device with a speaker, microphone, and a moderate amount of computational power should be able to use sonar; this includes cellular phones, PDAs, kiosks, and more.

This chapter has supported the first half of the thesis statement: that acoustic sensing can be used to sense the presence of computer users. The next chapter evaluates an application of this sonar presence sensing technique to display power management on laptop computers.

Chapter 3

Display power management

In this chapter we apply sonar-based user presence sensing to the problem of display power management (DPM) and we describe the first study of DPM policies. These policies control the mechanism of powering on and off the display—turning off the display typically reduces total system power by ∼31%. The most widely used DPM policy, *human interface device (HID) timeout*, powers off the display after a user-configurable period of human interface device inactivity, and powers it back on in response to activity. To increase energy savings, we also consider an alternative policy, *user presence detection*, that uses sonar sensing (as described in the previous chapter) to power off the display when user absence is detected. Our study captures the real-world performance both of DPM policies and of sonar presence sensing as observed for 181 users on different machines.

3.1 Introduction

Display power management (DPM) policies are autonomic control policies with which all computer users have direct experience. The goal of such a policy is to turn off the computer display whenever the user becomes inattentive and turn it back on again when the user becomes attentive. Toggling the display is a simple, but important, mechanism for controlling energy consumption. By display, we mean the LCD panel and its backlight (we do not consider the GPU).

LCD displays are major contributors to energy consumption. A past power measurement of laptops indicates that when dynamic voltage and frequency scaling (DVFS) is enabled on the CPU and the system is idle, the LCD (9%) and backlight (29%) together draw 38% of system power when fully brightened [MV05].

This idle condition is an extremely common state for modern laptop computers [ML91, Din99]. We present our own measurements of the potential energy savings (which exhibit similar behavior) in Section 3.3.

The DPM policy determines when to toggle on or off the display. The resulting energy savings are determined by user behavior, i.e., attentive and inattentive periods, and the effectiveness of the policy. Note that user attention is not usually precisely measured although a strictly optimal policy would require this information. There is an important tension in all realizable DPM policies: the more aggressive the DPM policy is in estimating user inattentiveness, the higher the energy savings, but also the higher the chance of annoyingly shutting off the display when the user is still attentive (e.g., reading a web page). We call such policy errors *irritation events*. A DPM policy must choose an operating point to resolve the tension between energy savings and irritation. In this chapter, we analyze how efficiently existing schemes make such trade-offs. Specifically, we answer the following questions:

- How well do current DPM policies work, both in terms of energy savings and user satisfaction?
- How much better could an optimal DPM policy do? How close to optimal are current policies?
- How can current DPM policies be improved upon?

We address these questions by examining the measured results of a large real-world study of two policies. The policies are the *human interface device (HID) timeout* policy that is universally employed in current operating systems, and a policy we developed, *user presence detection*, that tries to improve upon HID timeout, and that could be used in conjunction with it. We describe these policies in detail in Section 3.4.

Our measurements are based on a large-scale user study of 181 volunteers, each with his own machine, who collectively provide us with 3,738 hours of computer usage logs. We describe our study in detail in Section 3.6. We are aware of no prior work that evaluates any DPM policies in practice: this study is our main contribution.

In summary, our data show that:

- In practice, DPM policies can reduce display energy consumption by 81% at most and total system energy consumption by 25%.
- The widely-used HID timeout policy is surprisingly effective in practice, reducing display energy consumption by 51% and total system energy consumption by 15.3%. Furthermore, the median rate of irritation events for this policy is low: on the order of one per day.
- Evidence suggests that 44% of users do not customize the HID timeout parameter. This may result in those users having higher irritation rates than otherwise possible.

- User idle periods follow a power law distribution with little temporal correlation, suggesting that a simple predictor could be used to estimate idle period length.
- A combined policy using both HID timeout and presence detection could reduce display energy consumption by 10% compared to the HID timeout policy alone.
- The performance of sonar is quite sensitive to the available audio hardware and operating conditions. 30% of machines in our study were capable of generating and recording significant levels of ultrasound.

3.2 Related work

We are aware of no prior work that evaluates DPM policies in practice. The most relevant work studies user interaction with computer systems. The seminal work in that fold is that of Muttka and Livny [ML91], who studied system idle times, finding that desktop workstations, even in 1990, were idle much of the time. More recent work [Din99, Bea09] confirms that this remains the case today. Unfortunately, the traces collected from such studies cannot be used to analyze DPM policies since they are too coarse grained, and/or do not capture HID events.

There appears to be considerable opportunity for DPM policies to reduce overall system power consumption. Mahesri and Vardhan's 2005 study found that the display drew 38% of full system power on an idle laptop computer [MV05]. Roberson et al. studied the energy consumption of computer displays for the Energy Star program in a set of extensive office surveys [RHM+02, RWM+04]. Beyond powering off the display, which has for many years been one of the primary means of reducing energy consumption on deployed machines, Ranganathan et al. describe new display hardware with highly-adjustable reduced power consumption modes [RGMN06].

There have been proposals for new DPM policies based on user attentiveness. Dalton and Ellis proposed using a webcam to detect attention for DPM, but their evaluation was incomplete [DE03]. Visual attention and presence detection techniques are certainly promising; however, further work is needed to determine both their real-world effectiveness and their sensing and processing overheads. The user presence detection policy we evaluate in this chapter builds upon a previous study in which we showed that user presence can be detected using computer audio hardware under controlled laboratory conditions [TDDM09]. We have now demonstrated an application of the same technique and evaluated it in real-world environments. Our work takes place in the context of the Empathic Systems Project [DMD+07], which has generally found that there is a high degree of variation in user satisfaction with any given operating point in a system,

description	CPU type (Intel)	LCD size	CPU idle				CPU busy	
			LCD off	LCD dim		LCD max		
desktop examples:								
Dell Vostro 410	E8400 3 GHz	22"	63 W	92	(34%)	92 (34%)	128	(25%)
laptop examples:								
Lenovo T61	T7500 2.2 GHz	14.1"	13	14	(7%)	19 (32%)	29	(21%)
Dell Inspiron 8600	Pentium–M 1.6 GHz	15.4"	18	22	(18%)	26 (31%)	37	(22%)
IBM Thinkpad 240	Celeron 300 MHz	10.5"	7	8	(12%)	10 (30%)	16	(19%)
average contribution of LCD to total laptop power:					**12%**	**31%**		**21%**

Figure 3.1: Total system power consumption in Watts for various system states. Fraction of power due to the LCD is reported in parentheses. Note that desktop LCD backlights typically cannot be dimmed.

and that variation can be exploited by autonomic control that tailors for the individual user. Of relevance to this chapter, my collaborators have demonstrated a range of user-driven systems for power management [SOM+08, SPS+08, LMD+09, MCD+08]. However, none of these studies analyze display power management.

3.3 Display assumptions

DPM policies are applicable to both laptop and desktop computers. The obvious motivation for DPM on laptops is to lengthen battery life. Desktop computers have larger displays than laptop computers and therefore their displays consume more energy. Thus, DPM on desktop computers is important from environmental and economic perspectives.

The most recent measurements of display power we are aware of were published in 2005 [MV05]. To confirm and expand upon those measurements, we measured the total system power of several machines using a clamp ammeter on the AC supply line. By measuring the total electrical current with the LCD display on and off, we can readily calculate the current (and thus power) demand of display. We considered one desktop and three laptops, all running MS Windows.

Figure 3.1 shows the machines' total system power in several LCD and CPU load states. In the CPU busy state, the OS-reported CPU utilization was ∼100%. We also considered a CPU idle state, which is the common case, as reported in the literature we described in Section 3.2. Three LCD states were considered: fully off, dim, and full brightness. As we can see from the figure, the display's contribution to system power ranges from 7% to 34%, supporting earlier measurements. Note further that in a very common case, the CPU being idle and the display being bright, the display consumes 31% of system power, and this does not vary much across the machines we tested.

In the remainder of the chapter, we report reductions in display energy consumption and system energy

policy	attention estimate	failure modes
optimal	true attention	none
HID timeout	input activity	user reading without giving input
presence detection	physical presence as measured by sonar	user present but distracted, sonar errors in detecting presence

Figure 3.2: Attention estimates for each DPM policy.

consumption as percentages. For the display, a reduction in energy is equivalent to the change in duration that the display is powered on. For a reduction in system energy consumption, we normalize by the 31% figure, making the assumptions (again, supported by previous studies) that the idle state is most common and that that the display is at full brightness. Two examples where these assumptions approximately hold are web browsing and basic office applications. Note that there is no significant energy cost or delay involved in powering up or down the display. We ignore any wear-out costs associated with increased display power switching.

3.4 DPM policies

We considered two DPM policies in our study. The most important aspect of a DPM policy is how it estimates user attention. We summarize how our two policies do so in Figure 3.2. A comparison of the two policies can be considered a comparison of two attention estimation models.

3.4.1 HID timeout

By far the most commonly used policy today is *human interface device (HID) timeout*. HID timeout is familiar to most computer users; it powers down the display after some prescribed time interval has elapsed since the last HID (keyboard or mouse) event. HID timeout has a model of user attention with one parameter: the timeout length T. Under this model, the user is deemed attentive if the time since the last HID event is less than T and, conversely, inattentive if greater than T. HID input resets the interval and powers up the display.

This model works in some cases because it captures engagement between the user and computer. However, attention is not well measured: input activity-based techniques are unable to distinguish between a truly inattentive user and one who is actively reading the display without using the mouse or keyboard. Thus, the model parameter T is typically set to a conservative value: at least five minutes. Microsoft Windows' default value is five minutes for battery-powered operation.

3.4.2 User presence detection

The proposed user presence detection policy attempts to use sensor-based measurement of user presence as a user attention estimate. In the previous chapter, we showed how sonar can be used to detect user presence on a standard laptop computer without any additional hardware. The laptop computer emits a sound "ping" from its speaker and records the resulting echo on its microphone. In this study, the ping is a 22 kHz sine wave. This frequency is chosen to be high enough to be silent to humans (thus it is *ultrasonic*), yet low enough that it might be produced and recorded by the audio hardware built into most laptop computers. To sense the user, the computer can determine the amount of variation in the ping's echo. Little variation indicates a still room; more variation indicates motion in the environment, which may indicate the presence of a human. In the previous chapter, we showed that the technique is very effective under controlled laboratory conditions.

We use a threshold to determine whether a sonar reading indicates user presence or absence. High sonar reading variation corresponds to presence and low variation to absence, but calibration is required to determine a threshold value. In Section 2.5, sonar readings were taken while the user was known to be present and also while absent and a threshold value between these two readings was chosen. This calibration procedure works fine if the acoustic environment is static. However, if the computer is moved between different environments or if the environment's acoustics change then the threshold may no longer be set correctly.

Calibration is one of several practical issues that arise in real-world sonar presence sensing; we have addressed these as follows. It is easy to record sonar readings representative of a present user: for example, readings can be taken while the mouse is is motion. In order to avoid requiring the user to leave the computer to calibrate an absence sonar level, we approximate absent sonar readings as zero and simply set the threshold to one half of the present value. Each hour, the system is re-calibrated to choose an appropriate threshold.

Recall that we defined an irritation event as an instance when the user moved the mouse or pressed a key within five seconds of the display shutting off. Such irritation events are feedback from the user indicating that the latest presence detection decision was incorrect. To prevent repeated irritation, a factor of 0.8 is applied to lower the threshold after each irritation event. In this way, sonar-based display power management adapts to become less aggressive.

Sonar adapts to changes in the speaker volume levels by normalizing the reading to the recording's average ping intensity. Sonar uses only a narrow ultrasonic frequency band and it does not prevent normal use of the speakers, for example in playing music. In informal laboratory tests, playing music had no noticeable

effect on sonar performance.

3.5 Implementation

We employ both the HID timeout and user presence detection policies within a single piece of software, the Sonar Power Manager, a utility that we have designed for use with Windows XP, Vista, and 7, as well as Linux. Sonar Power Manager is designed to simultaneously test both policies by recording the actions of both policies in parallel.

If the sonar presence sensor indicates that the user is absent the display is shut off. Similarly, if the input activity timer expires the display is shut off. The timeout value is set equal to the display sleep timeout set in the Windows control panel for battery-powered operation. Thus, the presence detection policy adds an additional display shut-off criterion to the existing Windows HID timeout policy. By design, it is more aggressive than the default policy.

The Sonar Power Manager is a user-space utility implemented in C++ with the WxWidgets and PortAudio libraries to achieve portability across Windows and Linux. The software is open source and has been released with a permissive license. Both the source code and a Windows executable can be downloaded from our web site `http://empathicsystems.org` or `http://stevetarzia.com/sonar`. The website also describes the software's operation in more detail.

The Sonar Power Manager uses minimal power and CPU time. On a Lenovo T61 laptop, sonar sensing consumes ∼3% of the available CPU cycles on one core. The measured system power overhead due to sensing for the machines listed in Figure 3.1 was between 4% and 10%.

3.6 User study

We conducted a user study to collect usage traces appropriate for evaluating DPM. Our goal was to evaluate the timeout and presence detection policies for all users; we did this by running the policies simultaneously.

Recruitment and users To assist recruiting participants, a Northwestern University press release on our work led to a posting on the popular computer and technology news website `slashdot.org`. Shortly thereafter, the Sonar Power Manager was downloaded over 10,000 times. It is from these downloaders that we draw our data. Hence, it is important to note that our user sample represents those with an interest in technology news and may therefore may react differently to the HID timeout and user presence detection

policies than would the general population.

Sonar Power Manager collects log data on the operation of the two policies it tests. Users who installed the utility were given the option of "opting-out" of sending these logs to us. The majority of users opted-out of logging. It is also important to note that users who did opt-in ran the software for varying amounts of time. A user could also opt-out of logging at any time, and the logging process was automatically disabled after one week. We ignored logs from those who used the software for less than one hour.

Log contents For those who opted-in, Sonar Power Manager collected logs of sonar and power management events. Logs were compressed and sent to our servers every hour. The following information was time stamped and logged:

- The starting and ending time of HID input activity.
- Irritation events for both policies. Specifically, an irritation event is defined as a HID event that causes the display to wake less than 5 seconds after it was slept.
- The value of each sonar measurement.
- Times at which logging starts and stops.

Recall that Sonar Power Manager collects log data for both the HID timeout policy and the user presence detection policy simultaneously.

High-level view of the data-set We acquired 3,738 hours of usage by 181 volunteers, each with a different machine. There were 177 Windows users and 4 Linux users. Users ran the software for varying amounts of time. Figure 3.3 gives the distributions of the interval of time that the software was installed and the amount of time the software ran with full logging. The difference between these two durations is the time the machine was off or the user had logging disabled. The median amount of full-logging time was seven hours.

3.7 Results

We analyzed the logs collected during the user study to make the observations given in Section 3.1. We present our results in several parts: a characterization of user active and idle periods, an analysis of the effectiveness of the HID timeout policy, an analysis of the upper bound of energy savings for DPM policies based on user behavior, an analysis of the effectiveness and trade-offs of the user presence detection policy, and a characterization of sonar in practice.

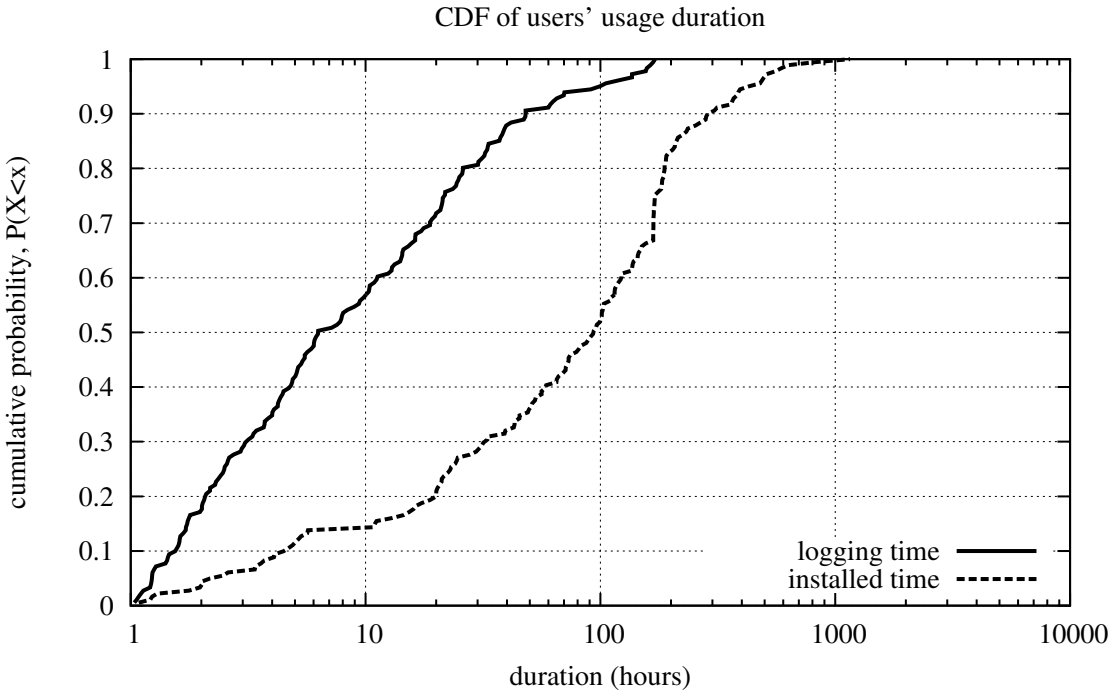


Figure 3.3: CDF of installed time and logging time. The difference is the time that the computer was off or logging was disabled.

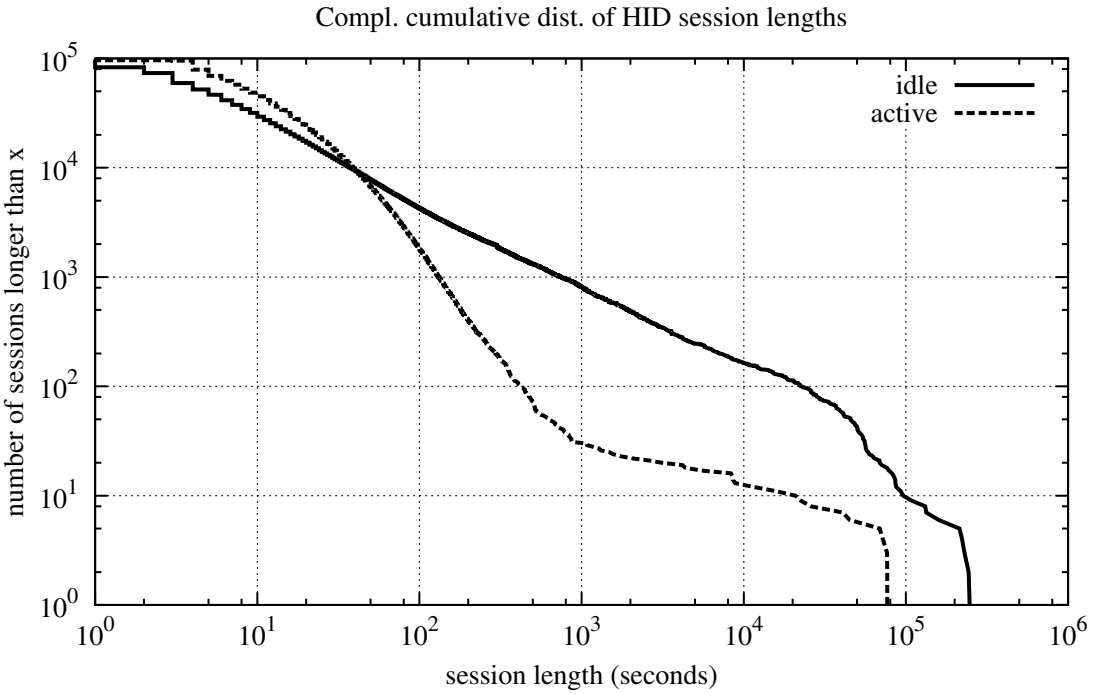


Figure 3.4: Distribution of lengths of HID-active and HID-idle periods.

In our treatment, we generally refer to two kinds of results: *aggregate* results are computed after first joining data collected from all users, while *individual* results are first computed for each user and then afterwards averaged across users. Aggregate results weigh each observation equally while individual results weigh each user equally. Since users provided different amounts of log data, these two averaging methods give different results. We also use logarithmic plots because it is well known that human perception of time durations scales logarithmically [EN81, Kom97, NL95].

3.7.1 Characterization of idle/active periods

Computer usage may be broken into a sequence of alternating active and idle periods, which we identify in our traces based on HID events. We consider a HID-active period as one in which HID events are spaced by no more than than one second. HID-active periods are separated by HID-idle periods which we define as those where more than one second elapses with no HID events.

Figure 3.4 shows the distribution of idle and active periods as defined above that were found in our data. Notice that the idle period distribution clearly follows a straight line in this log-log plot, indicating that it is a power-law distribution. Active periods tend to be shorter than idle periods and they seem to follow a

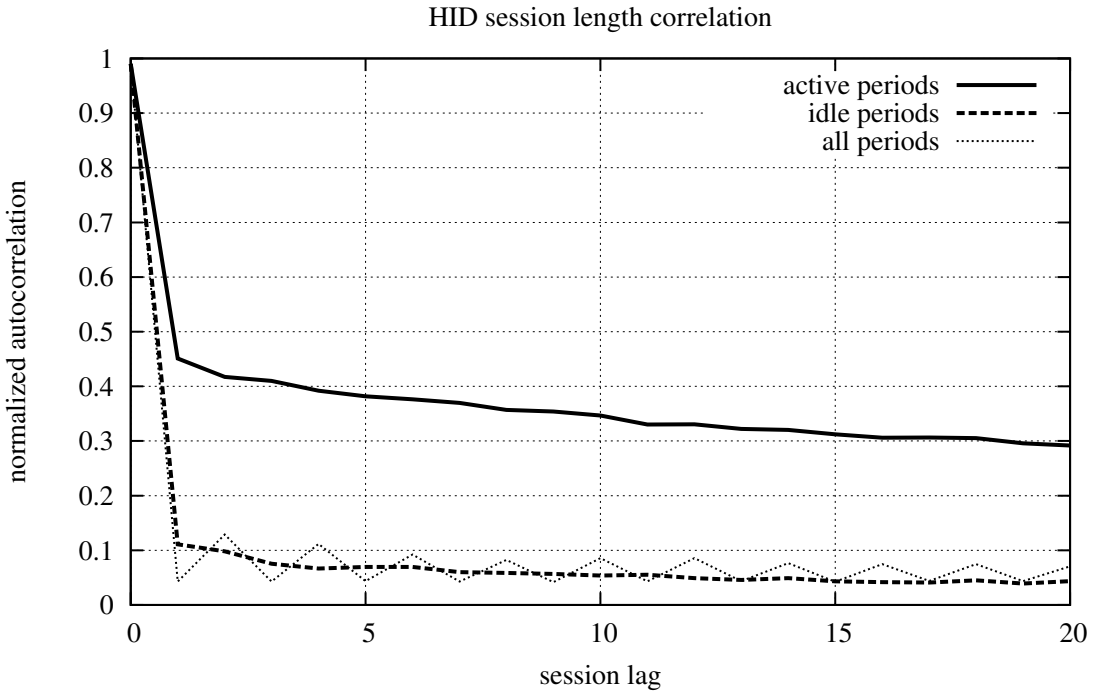


Figure 3.5: Autocorrelation of consecutive active and idle periods. These are individual results.

piecewise power-law distribution (note that the power-law exponent changes around 1000 seconds, evidenced by a change in slope).

In Figure 3.5 we show an autocorrelation analysis of active and idle sessions, considering active→active, idle→idle and (idle+active)→(idle+active). Although there is significant correlation of active periods to themselves, there is little to no correlation that might support the prediction of idle periods. This suggests that to predict an idle period's length, the best approach is probably simply to exploit the memory property of the idle periods' power-law distribution [HBD96].

3.7.2 HID timeout policy performance

Timeout setting distribution Figure 3.6 shows the distribution of users' DPM timeout settings. Given the observations that my collaborators have repeatedly made in the Empathic Systems Project (see Section 3.2), we might expect that users vary considerably in their performance expectations and that this should be reflected in a wide range of DPM timeout settings. This is indeed the case.

The peak of users at 5 minutes appears to be due to this being the default value in Windows. In Section 3.7.3 we will explore the implications of this peak. Figure 3.6 is also important since it shows the

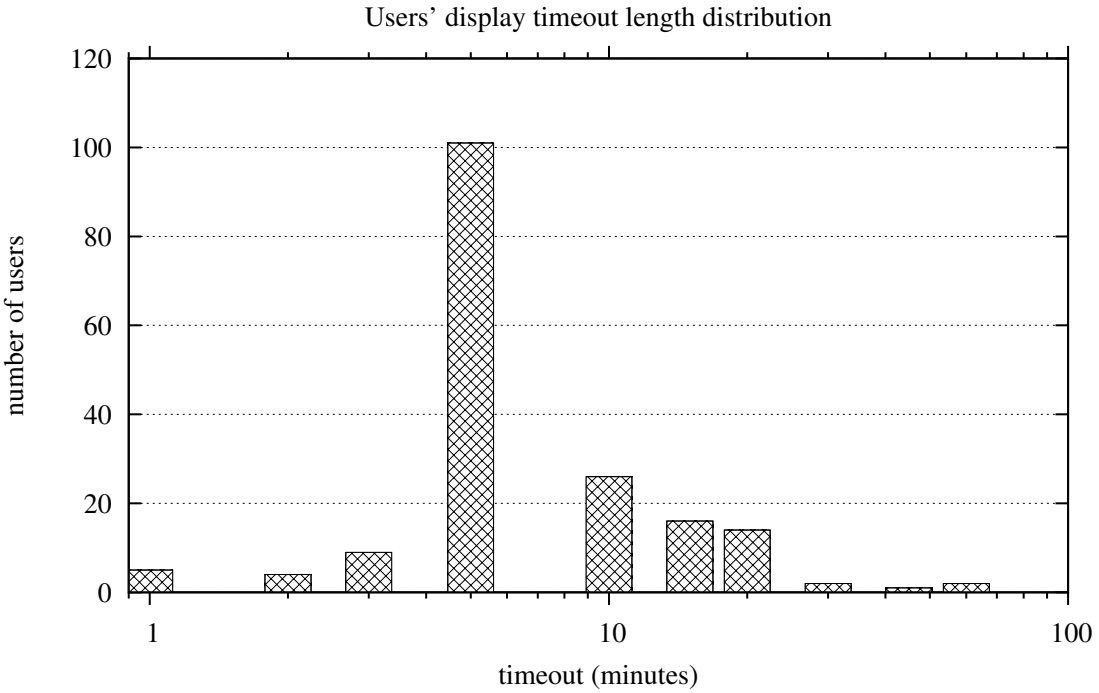


Figure 3.6: Distribution of users' HID timeout values.

distribution of samples, and thus the confidence levels, of the the results following in Figures 3.8 and 3.9.

Energy savings Figure 3.7 shows the distribution of the energy savings achieved by the HID timeout policy for users in our study. The savings is measured by the fraction of total runtime during which the display was shut off–called the *sleep fraction*. The median sleep fraction was 25%.

Figure 3.8 shows the sleep fraction as a function of users' timeout values. One might expect that the sleep fraction would decrease monotonically with the timeout value, but this is not the case. The mean aggregate sleep fraction across all users is about 51%. Although there there is some trend of lower energy savings with increasing timeout value, clearly the variation among users swamps this trend. Multiplying the display sleep fraction by the fraction of system power due to the display (which we approximate in Section 3.3 as 31%) gives the total system energy savings. The average total system power savings seen in our study is thus 51%×31%=15.8%.

It is important to note that the data of Figure 3.8 are based on users choosing their own timeout values. That is, it is an observational result, not intervention-based result. If we were to force different timeout values in a controlled way, the curve might be quite different.

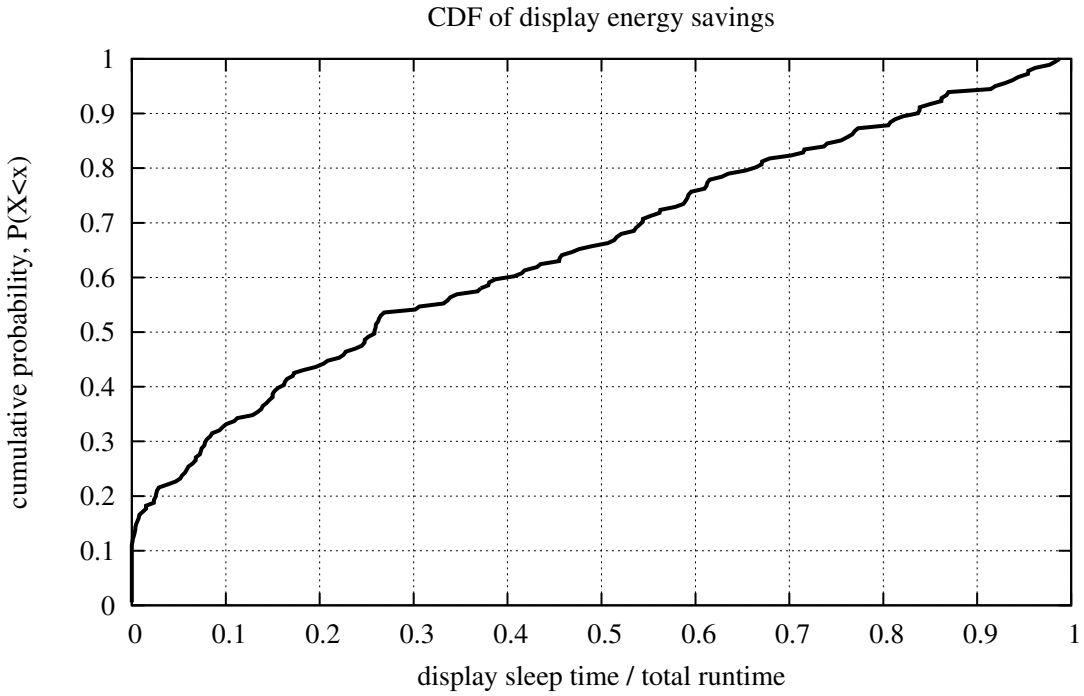


Figure 3.7: Distribution of users' achieved display energy savings under the HID timeout policy.

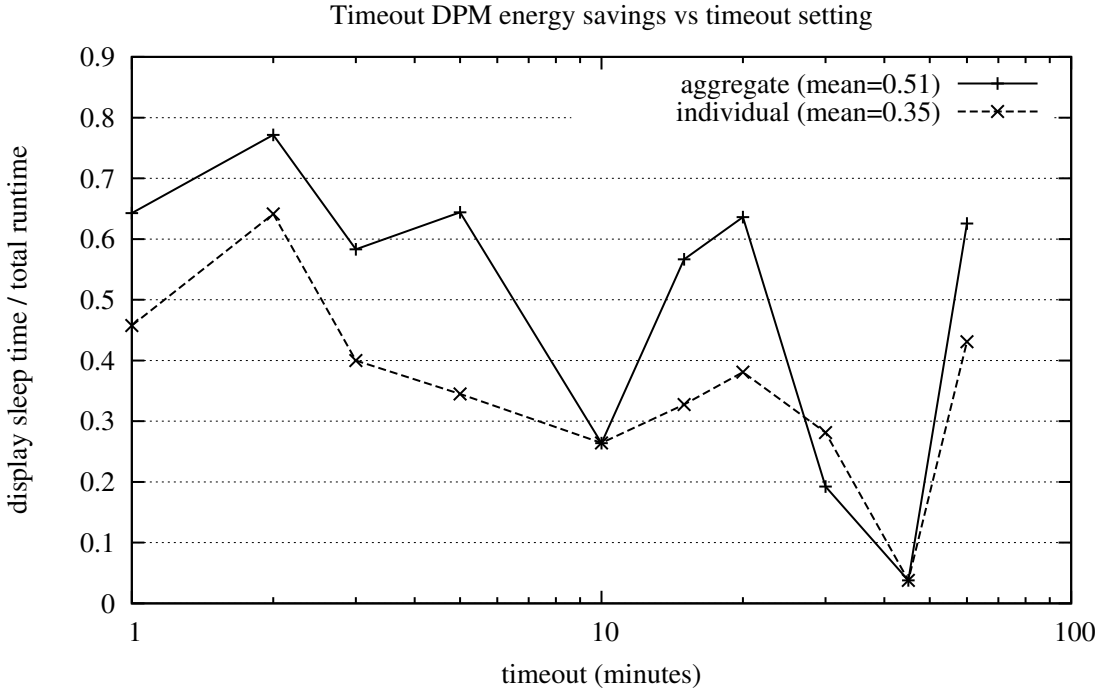


Figure 3.8: Display energy savings achieved as a function of the user's HID timeout parameter.

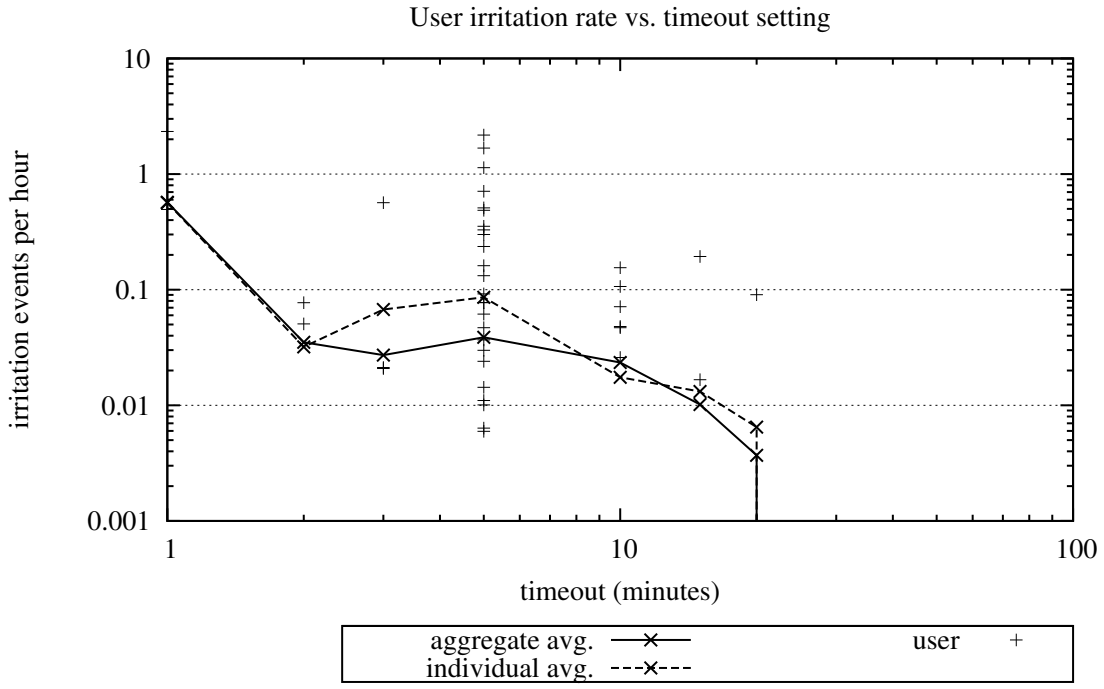


Figure 3.9: User irritation rate as a function of the user's DPM timeout setting.

Irritation rates User irritation occurs when the policy shuts off the display while the user is actually attentive. Of course, it is highly likely that the user will respond to such an *irritation event* with some HID input to wake the display. The sequence of the display being powered down quickly followed by HID input thus labels irritation events. In the following, we consider any display sleep period shorter than five seconds to represent an irritation event.

Figure 3.9 shows the user irritation rate as a function of the user's timeout setting. Note that for all timeout settings, except the lowest setting (one minute), irritation rates are quite low, on the order of once per day. We discuss the surprising peak in the curve at five minutes next.

3.7.3 Default users

Surprisingly, Figure 3.9 shows that users who chose a timeout value of two or three minutes experienced less irritation than those with a higher timeout value of five minutes. Five minutes happens to be the Windows default value, and it is used by roughly half of the users (see Figure 3.6). We suspect that the peak in irritation at five minutes is due to a significant group of users who neglected to customize the timeout value, and thus have more irritation and/or lower energy savings than would otherwise be possible.

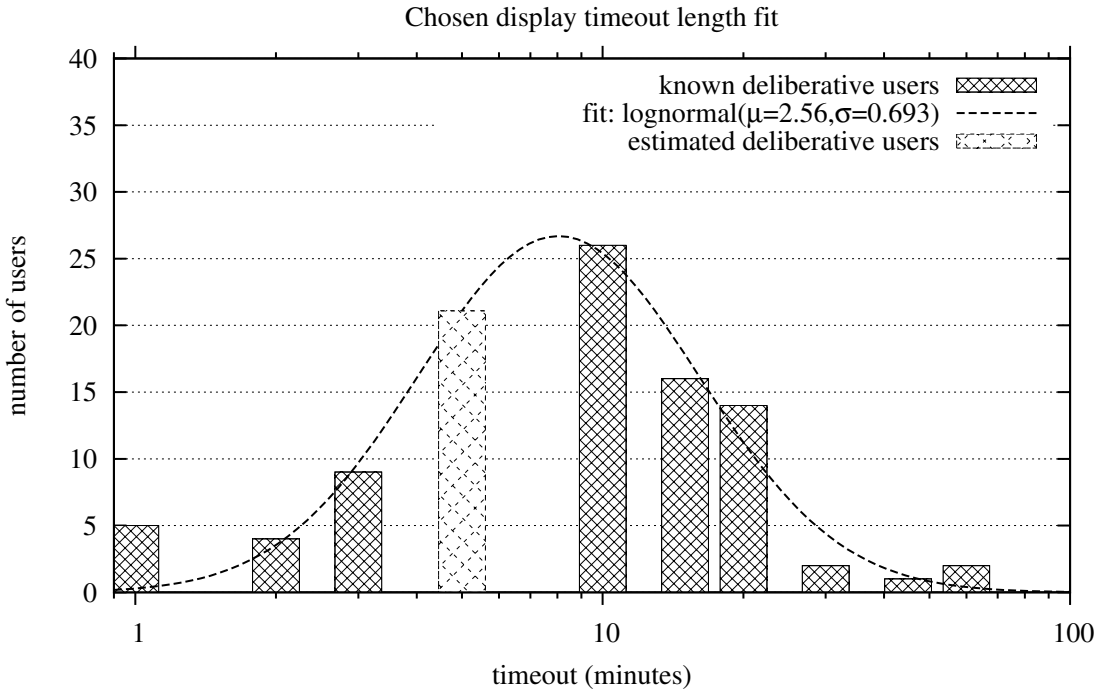


Figure 3.10: Distribution of deliberative users from Figure 3.6. The curve is fit to all users except those with the default five minute value.

Some portion of the five minute users have deliberately chosen five minutes, while the rest use it by default. We now attempt to estimate the sizes of these groups. We first ignore the problematic five minute users while fitting a distribution curve to the remaining users. Then we can use the distribution to estimate the expected size of the deliberative group. Figure 3.10 shows that a log-normal distribution appears to fit user timeout preference well, with PDF defined as follows:

$$PDF_{lognormal}(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left[-\frac{(\ln x - \mu)^2}{2\sigma^2}\right]$$

Using this fit, we estimate that, of the 101 users with the default five-minute timeout, 21% deliberately chose that value while 79% just use it by default. This latter group is significant in that it represents 44% of the users in our study. Due to the nature of our participants, it is likely that this latter figure significantly underestimates the proportion of users in the general population who ignore the customization controls for HID timeout policies. Assuming that the peak in irritation seen in Figure 3.9 is due to this population, irritation rates can be reduced by somehow forcing users to optimize the timeout setting.

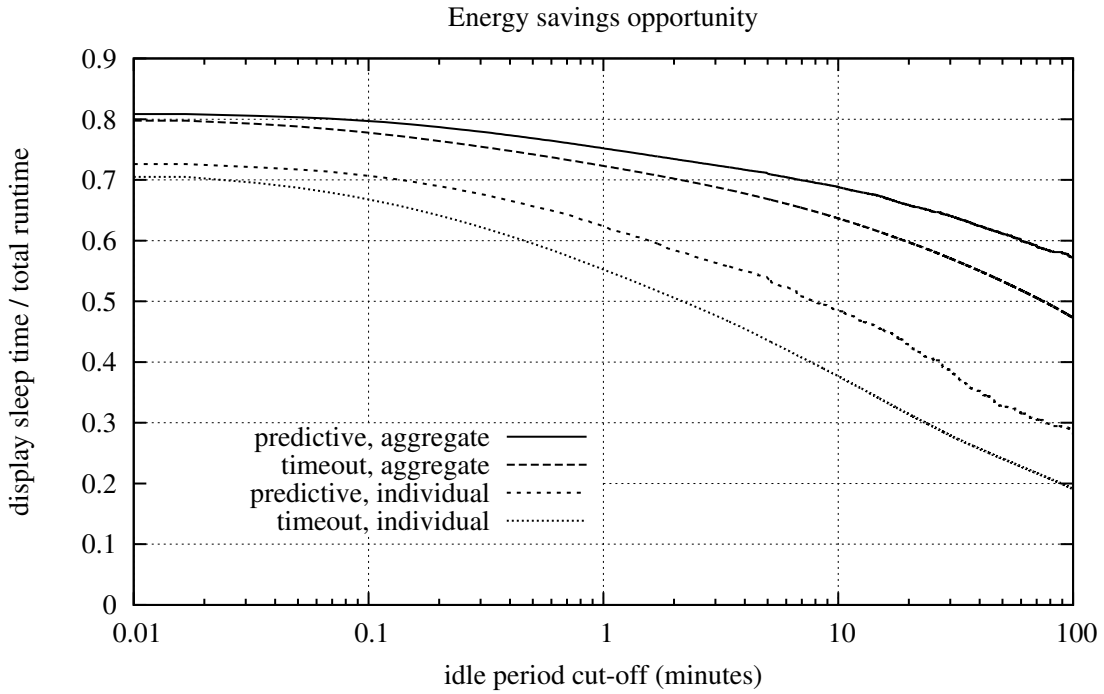


Figure 3.11: Energy savings opportunity as a function of the user attention cutoff. Predictive and timeout policies are compared, as described in the text.

3.7.4 Energy savings upper bound

Figure 3.11 shows the upper bound for display energy savings that any DPM policy can achieve. Here we assume that the display can be powered-down whenever there is no HID input for a given duration; we call this duration the idle period cut-off. We include all idle periods encountered in our data, not just those where the user was inattentive. Because of this, the upper bound is loose–a policy that captured actual user attention would have lower energy savings. At most, the display energy savings are 81%, for a system energy savings of 81%×31%=25%.

Two classes of policies are compared. The predictive policy assumes foreknowledge of an idle period's length and thus allows the display to be powered down for the entire duration of the period. The timeout policy powers down the display only after an idle period extends beyond the cut-off value. The closeness of these upper bounds suggests that using a sophisticated idle period length prediction scheme will give little benefit. Furthermore, the diminishing returns of targeting brief idle periods is seen as the curves flatten out below about one minute. Conversely, the steeper slopes around the default timeout value of five minutes indicate that energy savings are sensitive to the timeout setting in this regime.

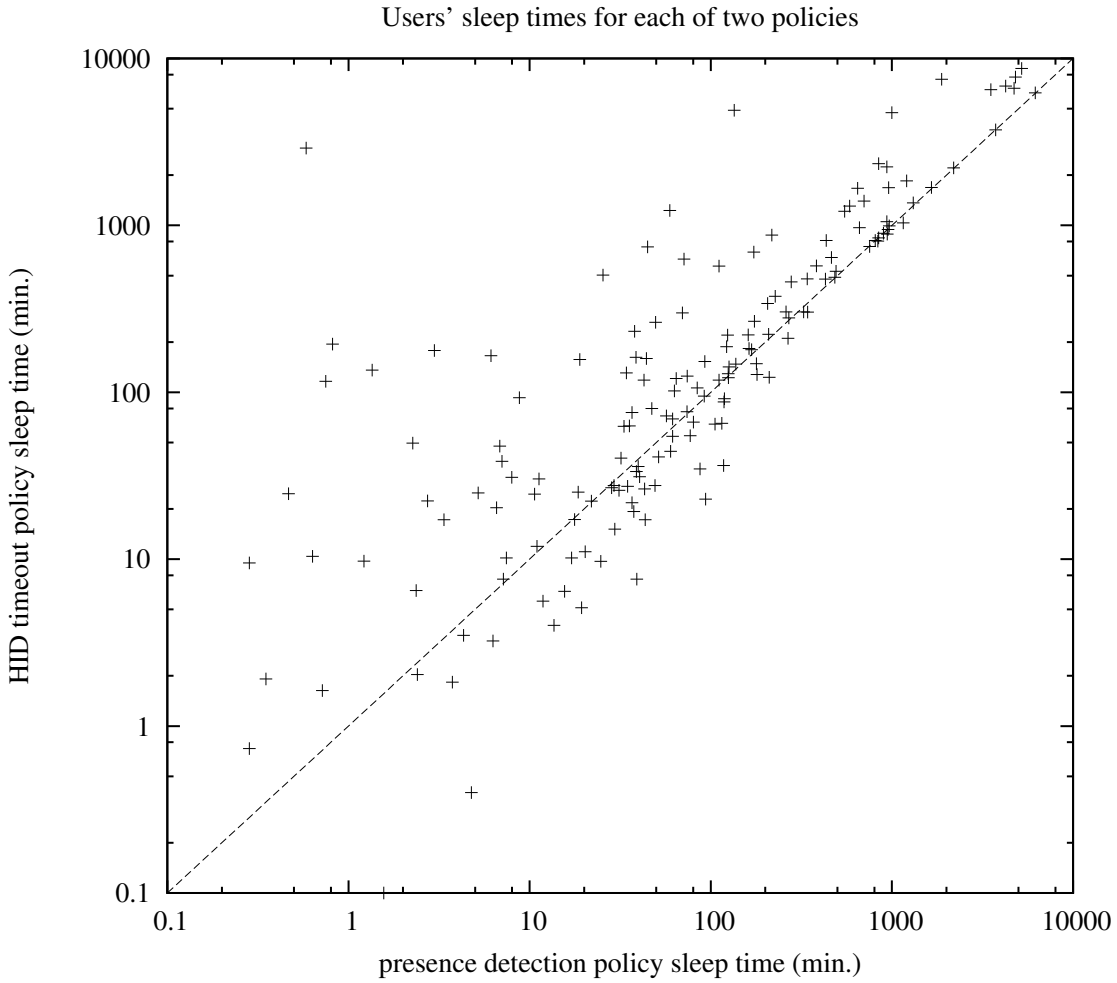


Figure 3.12: Comparison of energy savings due to presence detection and HID timeout policies. Each point represents a user and those below the diagonal line had more energy savings due to the presence detection policy than the HID timeout policy.

3.7.5 User presence detection policy results

We now show the benefit to DPM of adding a sonar presence detection sensor. Initially, we consider all users, but later we attempt to isolate the machines on which sonar presence sensing worked well.

Energy savings Figure 3.12 shows a comparison of the sleep times (and thus the energy savings) of the HID timeout and sonar presence detection policies when they were run in parallel. Many users (those above the diagonal) experienced more energy savings due to the HID timeout policy than presence detection alone. For the users below the diagonal, presence detection alone performed better than HID timeout.

We consider a simple combined policy in which both HID timeout and presence detection policies are run

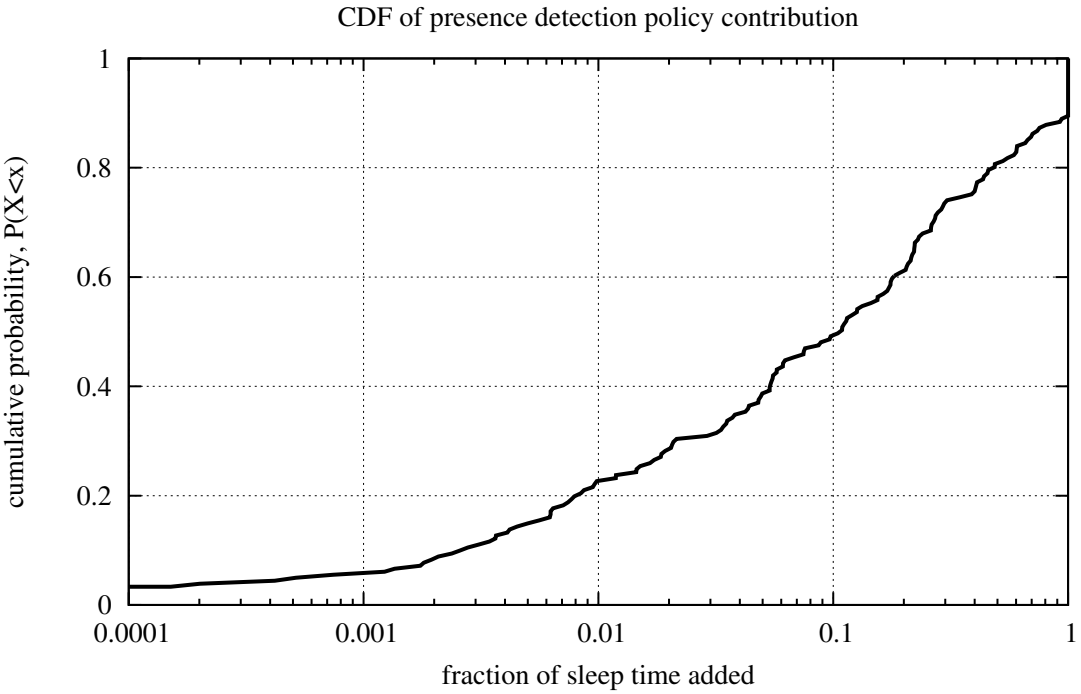


Figure 3.13: Additional display energy savings due exclusively to presence detection in the combined presence-timeout policy. This is the benefit of adding presence detection.

simultaneously. Either policy can decide to turn off the display, and HID input turns it back on. Figure 3.13 considers the additional energy savings that sonar provides in the combined policy. It shows the energy savings that would be lost if sonar presence sensing were disabled. In particular, the CDF shows that for half of the users, presence detection contributes at least 10% of the total display energy savings. Also, the presence detection policy at least doubled display energy savings for almost 20% of users.

Sensing overhead There is a small energy overhead associated with sonar measurement so we must judge whether the gains from incorporating sonar outweigh the costs. The energy cost is directly proportional to the number of sonar readings taken. The energy gain is directly proportional to the amount of time that the display can be slept due to presence detection. It is important that we do not count intervals in which the display would have been slept by the timeout policy; we are interested only in the additional sleep time beyond this baseline. Figure 3.14 shows the ratio of the gains to the number of sonar measurements. For example, sonar presence detection added an average of 0.12 seconds of display sleep time per sonar reading for the median user.

To compute the energy in absolute terms, we extrapolate the energy overhead based on the 7% average

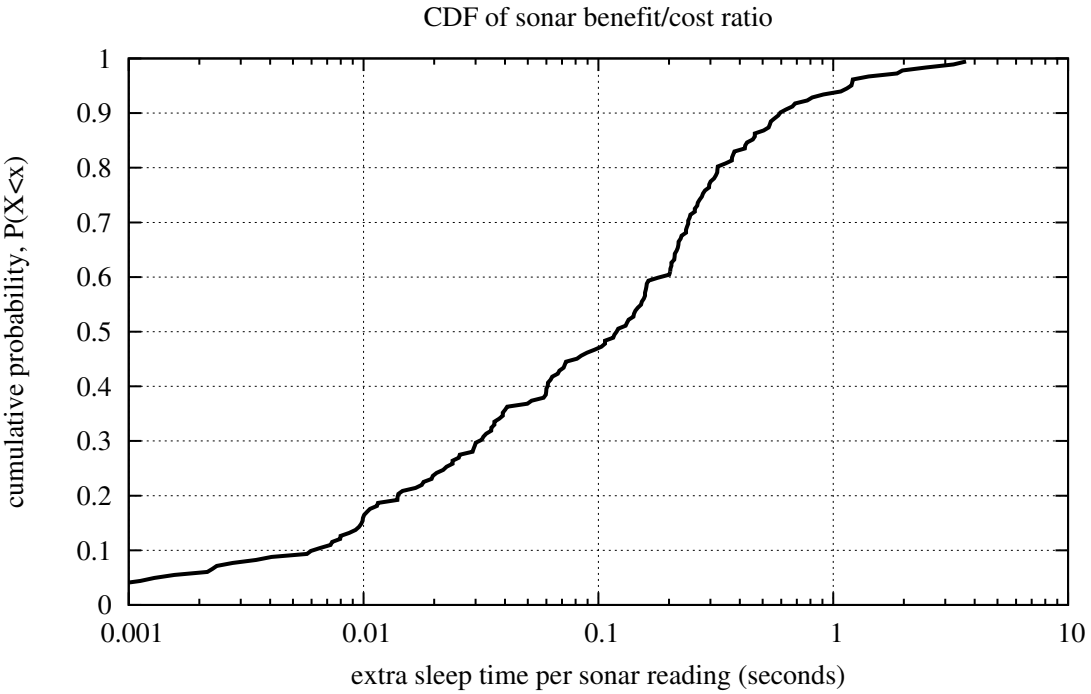


Figure 3.14: CDF of additional display sleep time per ∼1 s sonar reading.

sensing power overhead recorded on lab computers (see Section 3.5). In Figure 3.14, the top 10% of users gained at least 0.6 s of display sleep time per one-second sonar reading. Costs are about (7% × 1 s)/(31% × 0.6 s) = 38% of gains for these users. For users with less than 0.22 s of extra display sleep time per sonar reading, costs were at least at large as gains. Sonar can and should be disabled for these 67% of users.

Irritation rates While Figure 3.9 showed low user irritation rates for the HID timeout policy, Figure 3.15 shows more significant irritation rates for the presence detection policy. The presence detection policy's median value of one irritation event per hour is relatively low but significantly higher than the HID timeout policy's rate of one per day. The combined policy's irritation rates track those of the pure presence detection policy. The curve annotated "combined – good users" shows the performance of the combined policy for situations where it is most effective, as discussed next.

Figure 3.16 shows that there is some correlation between energy savings and irritation rates for the combined policy. As expected, at the left side of the plot, we see that users for whom sonar rarely powered off the display were rarely irritated. Note that some users with high energy savings rates did not experience high irritation rates (users below the dashed line in Figure 3.16). These are the users for whom sonar provided a real benefit over the pure HID timeout policy.

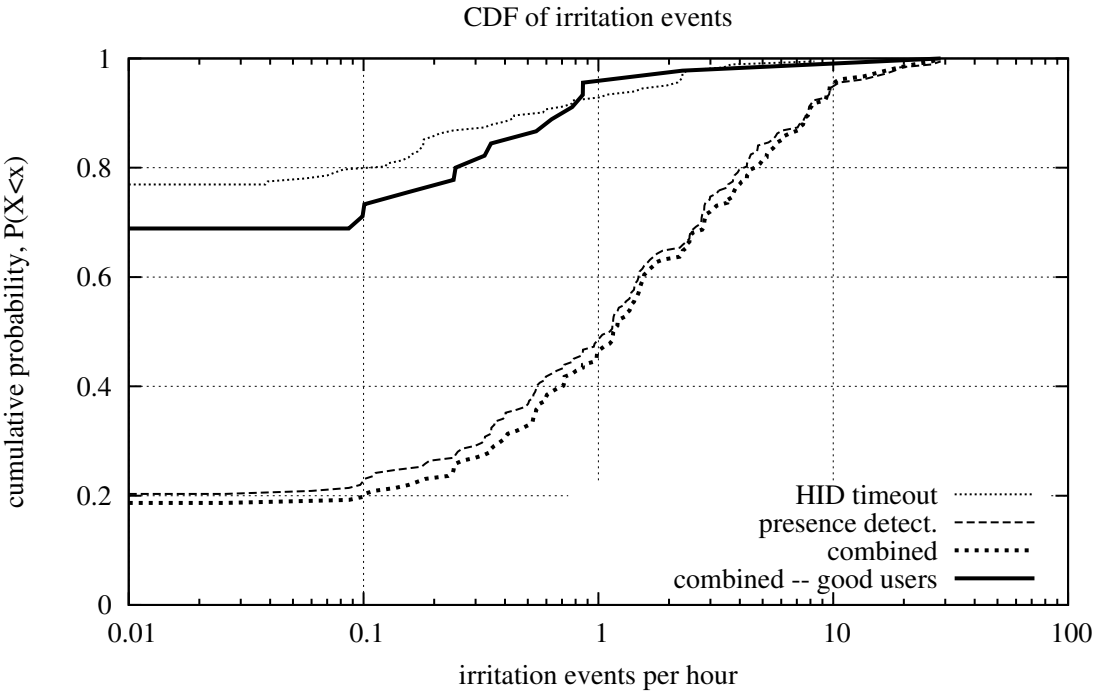


Figure 3.15: CDF of irritation events per hour.

Thus far, we have explained a combined HID timeout and presence detection policy that simply runs the two in parallel and allows either to turn off the display. A more advanced policy would determine when presence detection is likely to be effective. One approach would be simply to run the two policies to collect information similar to that of Figure 3.16. The user would then choose a dividing line (as shown in the figure) to trade off between irritation and energy savings. The example line represents a trade-off that requires fewer than 2 irritation events per hour of additional display sleep time. The "combined – good users" curve in Figure 3.15 reflects the effects of the example line. Another combined policy might track the irritation rates of the two policies and choose the one that currently has the lowest rate, over some window. This would be an application of the multiple experts strategy [BB97].

3.7.6 Characterizing sonar

Our study also allowed us to characterize the performance of ultrasonic sonar on ordinary computers in ordinary situations. The previous analysis, in Figures 2.9 and 2.11, focused on a small range of audio hardware under controlled lab conditions. The following results cover a wide range of hardware in real usage environments.

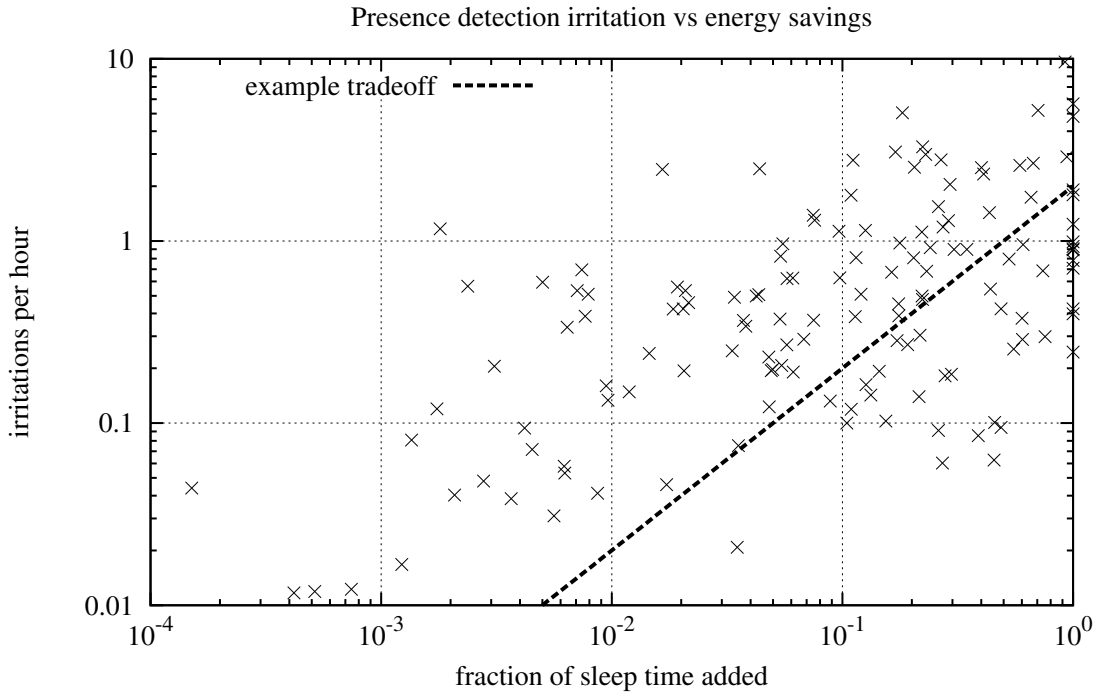
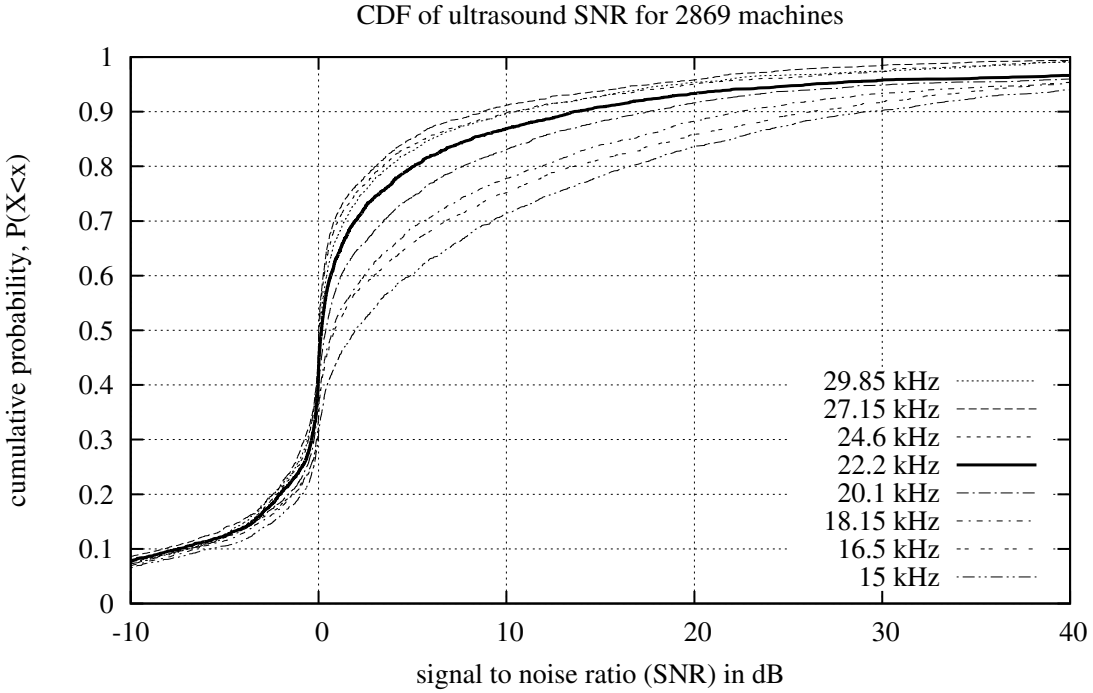
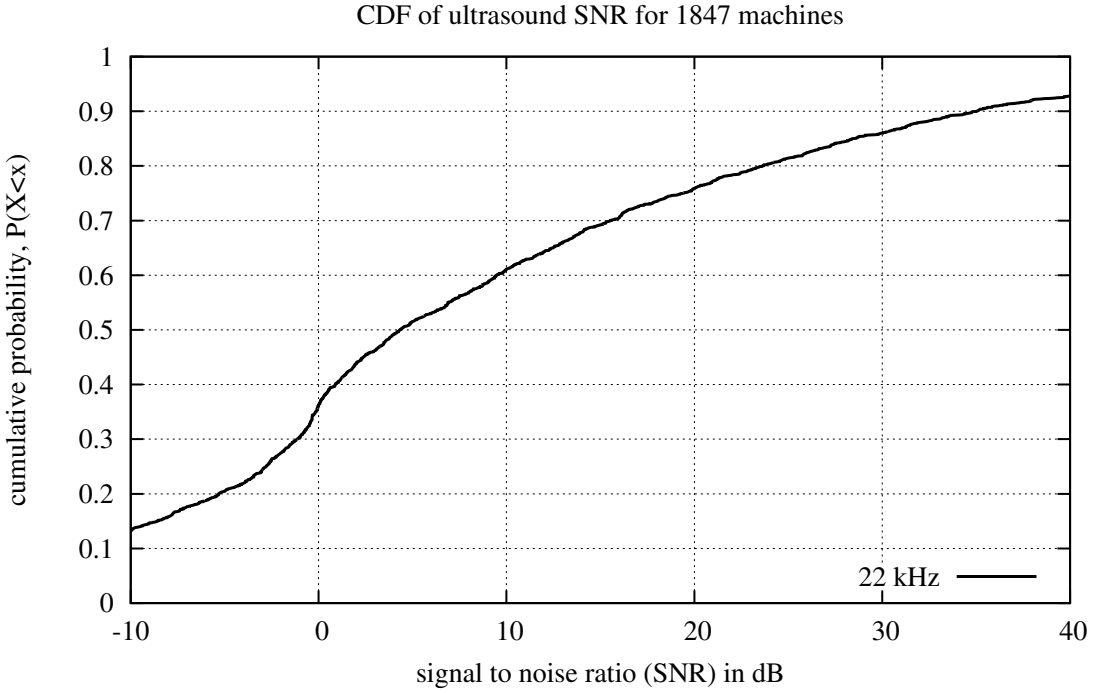


Figure 3.16: Relationship between irritation rates and energy savings for the presence detection policy. The 44 users below the dashed line experienced fewer than 2 irritation events per hour of sleep due to presence detection.



(a) SNR spectrum calculated from white noise stimulus.



(b) SNR calculated from a single sine wave.

Figure 3.17: Ultrasound signal to noise ratios (SNR). On a given machine, a low SNR means that sonar will not work.

Sonar-based detection of user presence requires audio hardware that is capable of both emitting and recording ultrasound. Sonar Power Manager was configured to conduct a calibration phase and report its findings back to us. Because calibration happens at startup time, we have calibration data from an order of magnitude more users and machines than we have long-term logging data. The important part of the calibration data is a measurement of the ultrasound signal to noise ratio (SNR). This is accomplished by generating and simultaneously recording ten seconds of white noise in the range 15 to 30 kHz, as well as recording ten seconds of silence. The white noise is the desired signal and the silence is the background noise. Their power ratios at different frequencies give an approximate SNR spectrum, for which we show the CDF for 2,869 machines in Figure 3.17(a). Figure 3.17(b) shows the SNR calculated from recordings of a 22 kHz sine wave. This measurement occurred after the previous one, and so there is an attrition of users—only 1847 machines are included. These two methods gave significantly different SNRs for 22 kHz. We focus on the results from the sine wave emission because it matches the implementation of our sonar system. The advantage of the white noise method is that it allows all frequencies to be measured at once; however, I suspect that it's results may be skewed by spectral leakage and interference among the frequencies.

Figures 2.9 and 2.11 in the previous chapter show that, in the lab, good presence detection performance was achieved with hardware having an SNR above 15 dB. As expected, lower frequencies give a generally higher SNR at the expense of becoming audible to more humans. At the frequency used by the experimental software (22 kHz), Figure 3.17(b) shows that 30% of machines had a good SNR. We might expect users with high irritation rates seen in Figure 3.15 and low cost-benefit ratios in Figure 3.14 to fall into the 70% of machines with low ultrasound SNR. However, this did not appear to be the case. Figure 3.18 shows correlation between SNR and both irritation and energy savings. Strangely, a clear correlation is not evident. Based on these results, it seems clear that any combined sonar and HID timeout policy should select between the two using the performance of the overall system, rather than the SNR found during calibration. Ideally, an ultrasound calibration technique capable of predicting sonar presence detection performance would be found.

We just stated that 30% of available machines were capable of generating and recording ultrasound with their current audio systems. While it straightforward to understand why a commodity microphone can capture ultrasound, it is surprising that speakers are capable of generating it. We attribute this to the lack of low pass filtering in common output paths combined with cheap speaker elements being quite small.

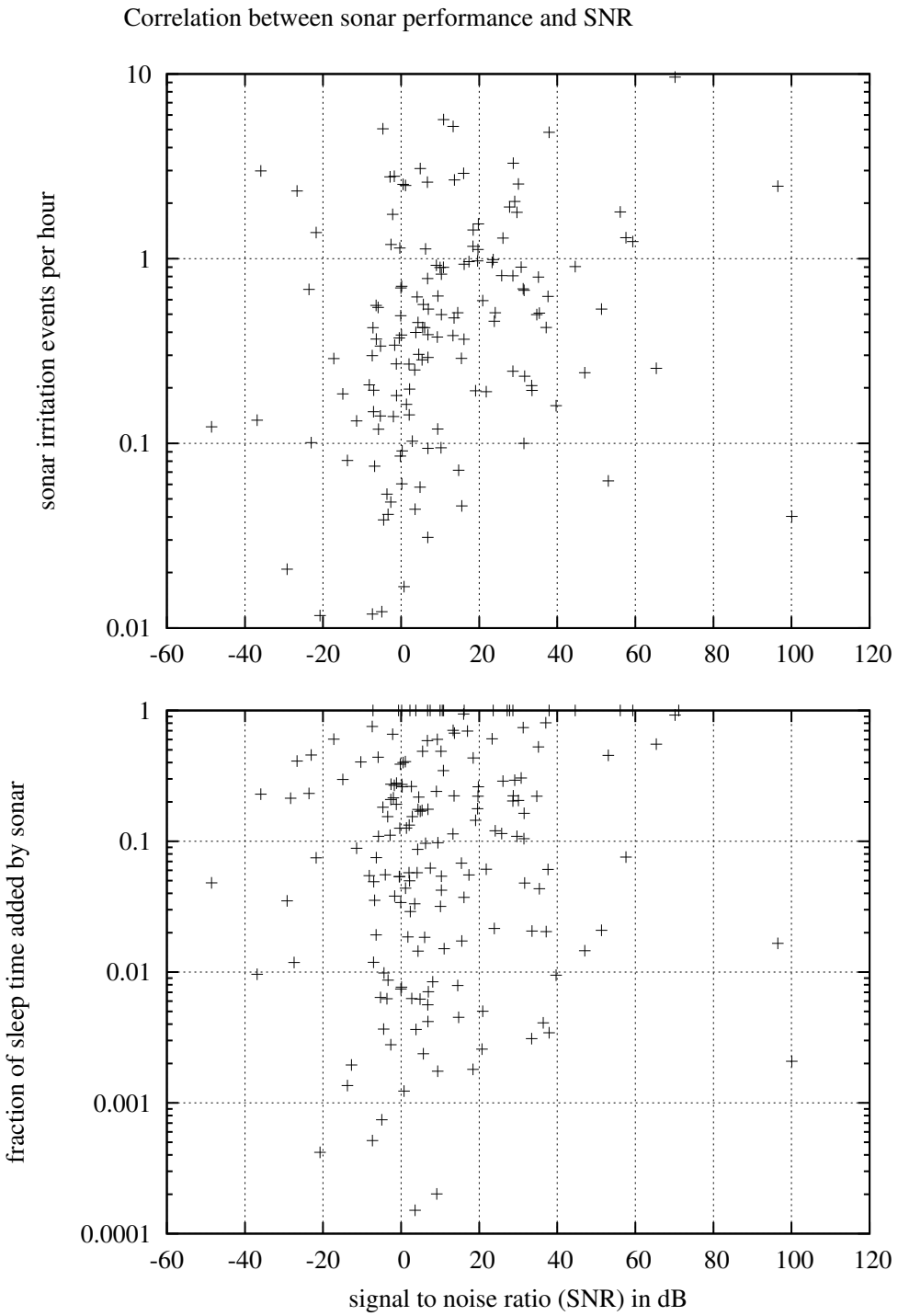


Figure 3.18: Correlation between SNR and sonar-induced irritation (top) and energy savings (bottom).

3.8 Conclusions and future work

We have presented the first study of the real-world behavior of display power management (DPM) policies, considering the widely-used HID timeout policy, a sonar-based presence detection policy, and a combined policy. We found that user idle periods follow power law distributions with little temporal correlation. The maximum possible reduction in energy used for the display is 81%, while the HID timeout policy manages to achieve 51%. Our results suggest that 56% of users have already customized the HID timeout policy on their machines resulting in surprisingly low levels of user irritation of around once per day. However, 44% of users have not, and these users experience more irritation. When the presence detection policy was combined with the HID timeout policy, half of the users gained at least 10% more display sleep time. 30% of available machines can generate and record ultrasound effectively.

Based on the experimental results, we recommend modifying Operating System DPM implementations as follows. We have described a method of detecting user irritation events caused by false display shutoff policy decisions. Whenever this happens, the user should be prompted to optionally lengthen the DPM timeout. This feedback scheme would allow users to easily fix a sub-optimal policy setting which might otherwise be ignored. Alternatively, a fully autonomic policy might increase the timeout length by, for example, 10% whenever an irritation event occurs while reducing the timeout by 10% each active hour. We also recommend adaptively adding sonar presence detection to the DPM policy. During runtime, if either the ultrasound SNR or achieved display energy savings are too low or the irritation rate is too high, sonar should be disabled.

So far we have considered only DPM policies for current hardware. Several additional paths exist. First, better user attention detection sensors could be employed, for example the biometric sensors we have begun to use to gauge user satisfaction [SPS+08]. Dedicated presence sensors (e.g., infrared motion sensors) can be both reliable, energy efficient, and inexpensive. Simply adding a display power switch would be an easy way to allow users to reduce the energy consumption of their laptop computers. Another research direction is DPM policies for emerging display technologies. In particular, e-paper displays consume power only when the display contents change. LED backlit displays (which are already replacing traditional fluorescent backlit displays) have lower overall power consumption. Some experimental displays have been demonstrated which allow only part of the display to be lit. The proliferation of new computer displays throughout homes and workplaces might provide an additional motivation for DPM in reducing visual clutter/confusion as well as saving power.

The chapter concludes my exploration of acoustic sensing of user presence. The next chapters describe a second new acoustic sensing technique.

Chapter 4

Location

This chapter describes a new technique for determining a mobile phone's indoor location even when Wi-Fi infrastructure is unavailable or sparse. Our technique is based on a new ambient sound fingerprint called the Acoustic Background Spectrum (ABS). An ABS serves well as a room fingerprint because it is compact, easily computed, robust to transient sounds, and surprisingly distinctive. As with other fingerprint-based localization techniques, location is determined by measuring the current fingerprint and then choosing the "closest" fingerprint from a database. An experiment involving 33 rooms yielded 69% correct fingerprint matches meaning that, in the majority of observations, the fingerprint was closer to a previous visit's fingerprint than to any fingerprints from the other 32 rooms. An implementation of ABS-localization called Batphone is publicly available for Apple iPhones. We used Batphone to show the benefit of using ABS-localization together with a commercial Wi-Fi-based localization method. In this second experiment, adding ABS improved room-level localization accuracy from 30% (Wi-Fi only) to 69% (Wi-Fi and ABS). While Wi-Fi-based localization has difficulty distinguishing nearby rooms, Batphone performs just as well with nearby rooms; it can distinguish pairs of adjacent rooms with 92% accuracy.

4.1 Introduction

The goal of this chapter is to allow a basic mobile device, such as a smartphone, to cheaply and quickly determine its location to the resolution of a single room. The mobile device may not have any specialized hardware, and there should be minimal, if any, dependence on computational or network infrastructure.

More specifically, we seek a system that computes a

$$\textit{fingerprint} \rightarrow \textit{roomlabel}$$

mapping where *fingerprint* represents a measurement of the room the mobile device is currently in, and *roomlabel* represents a semantically-rich representation of the room. The notion of a room can be generalized to any physical place; in the next chapter we focus on hallway segments rather than enclosed rooms.

Fingerprint acquisition and mapping to room labels must be rapid. Fingerprints and room labels must be small so that it is possible to cache or prefetch them despite the device's limited memory. It must be possible to acquire a fingerprint and do the computation (using cached fingerprints) on the mobile device even if wireless communication is not presently possible. Above all, the fingerprint computation must be reliable in the presence of noise and environmental variations. In summary, a location fingerprint should be **DECENT**:

- **D**istinctive ..... (see Section 4.5.1)
- r**E**sponsive ..... (see Section 4.5.2)
- **C**ompact ..... (see Section 4.5.2)
- **E**fficiently-computable ..... (see Section 4.8.3)
- **N**oise-robust ..... (see Section 4.5.3)
- **T**ime-invariant ..... (see Section 4.5.4)

The Acoustic Background Spectrum (ABS) technique we describe here meets these requirements. The intuition behind ABS-based localization is that modern life is full of noises: computers whirr, lights buzz, and air conditioners blows. The field of architectural acoustics tells us that a room's geometry and furnishings strongly affect what is heard [Kut73]. That is, the persistent acoustic drivers of a room, and the room's impulse response combine to form a distinct "quiet sound" of the room. While it is obvious that a lecture hall might sound different than an office, we show that two otherwise similar-looking rooms are also very likely to have different persistent acoustic characteristics.

It is not clear whether humans are capable of distinguishing rooms by sound. My personal experience is that we generally ignore persistent background sounds; however, when making a conscious effort, people can recognize rooms by sound. I built a web-based listening test which allows you to test your own ability to recognize rooms by sound (available at `http://stevetarzia.com/listen`). However, user performance statistics were not gathered.

Using ABS, which is implemented in our publicly available Batphone application[1], an Apple iPod is able to determine what room (and building, etc.) it is in using a compact 1.3 kB representation of the room's acoustic spectrum as a fingerprint. The fingerprint is acquired within 10 seconds using the built-in microphone and by the eleventh second a list of room descriptions appears, which is ranked by likelihood of being the current room. Users can easily contribute new (*fingerprint*, *roomlabel*) pairs from the application.

The problem of indoor localization has been well-studied, but no ideal, all-purpose solution exists. Techniques based on observations of signals from radio beacons (such as Wi-Fi/cellular base stations or GPS satellites) have shown the most promise. However, these techniques are hampered by the fundamental inconsistency of indoor radio wave propagation caused by changes in the physical environment and interference from other electronic devices. Even the number of users connected to a base station affects the observed signal strength. Various studies have shown that Wi-Fi signal strengths vary significantly throughout the day and that these variations are not uniform (the base stations vary independently) [OVLdL05, Figure 1] [HFL$^+$04, Figure 7]. Signals from cellular base stations seem to be much more temporally-stable than Wi-Fi, however they are much less densely spaced, leading to localization difficulties.

Despite these observation uncertainties, researchers have built Wi-Fi-localization systems capable of 95% room-level accuracy in ideal, static environments [HFL$^+$04, YA05]. Indeed, we replicate this superb Wi-Fi localization performance in Section 5.4.3 with our own implementation and experiment. However, real usage environments generally make the problem harder in one of two ways: base station density may be low or occupancy variations may cause long-term signal variations. In either case, the reported Wi-Fi localization accuracy falls below 70% [HFL$^+$04, Figures 6 & 8]. In addition, developing countries typically lack Wi-Fi infrastructure, but low-end mobile phones are widespread. Such devices have only a cellular radio and a microphone with which to sense the environment.

In contrast to radio-based techniques (Wi-Fi or cellular), our ABS technique works even if no radio signals are available if a sufficient collection of fingerprints is cached on the mobile device. Furthermore, the errors of ABS-based localization are different from those of radio-based localization, and thus the techniques can be combined for better performance than either one alone. In our experiments, the Batphone implementation of ABS improves the accuracy of the commercial Wi-Fi-based localization service used by the iPod from 30% to 69%. The nature of the errors for radio-based localization is different from that for ABS-based localization: radio-localization errors tend to confuse nearby rooms, while ABS-based localization errors are generally *not* geospatially clustered. For example, using ABS alone allows us to distinguish adjacent rooms with 92%

[1] Users of iPhones, iPads, and iPod Touch models with microphones can search for "Batphone" in Apple's app store. iOS version $\geq$ 4.0 is required.

accuracy. The consequence is that user knowledge of general location is likely to increase the probability of selecting the correct room from a top-k list when ABS-based localization is used.

There is prior work on using acoustics for localization; a detailed comparison is given in Section 4.2. SurroundSense is the closest to our work [ACC09]. It combines multiple sensors, including acoustic ambience measurement, visual ambience, and cellular-based localization to determine location. SurroundSense's acoustic fingerprint is a compact histogram of a sound recording from the time domain, while ABS uses a compact *spectral* representation. Furthermore, our focus is on exclusively sound-based localization, and the ABS technique provides dramatically higher performance than the histogram technique when only the microphone is available. Another work, SoundSense [LPL+09], uses spectral features to classify sounds observed on a mobile device. Our work differs in that it targets localization rather than activity detection. Accordingly, we focus on background sounds while SoundSense focuses on transient sounds.

In summary, our contributions are as follows:

- We introduce the Acoustic Background Spectrum (ABS) fingerprint, a new acoustic ambience fingerprint that is compact, easily computed, robust to transient sounds, and able to distinguish remarkably similar rooms. ABS allows us to build the first accurate localization system based solely on microphone sensing.
- We introduce the Linear Combination Distance for combining localization fingerprints which, when applied to ABS and Wi-Fi, dramatically improves indoor localization accuracy over each individual method.
- We describe a trace-based simulation system that allows us to study and evaluate different acoustic localization techniques, and different parameters for an acoustic localization technique, using the same set of real acoustic traces (recordings). ABS was developed using this system and a collection of 264 acoustic traces, which we have made publicly available.
- We describe and make publicly available Batphone, an open-source Apple iPhone implementation of ABS-based localization that demonstrates the technique's real-world performance and its low overhead. Batphone also includes the option to use Wi-Fi-based localization and combined ABS/Wi-Fi localization.

The rest of this chapter is structured as follows. Section 4.2 describes related work. Section 4.3 describes the new Acoustic Background Spectrum (ABS) room fingerprinting technique. The ABS technique requires that we select values for certain parameters. We did so through trace collection and simulation. Section 4.4 describes our traces and simulator, while Section 4.5 describes our study to determine appropriate ABS

parameters. This section also provides an initial evaluation of the performance of ABS and sensitivity to parameter variation. Section 4.6 describes the implementation of ABS within Batphone, including its integration with Wi-Fi-based localization. Section 4.8 evaluates the performance of Batphone. Section 4.9 concludes the chapter.

4.2 Related work

Indoor localization has been well-studied in the mobile computing community. A user may be carrying a sensing device (such as a mobile phone) or may be moving within a field of fixed sensors. In either case, knowledge of the user's position can be useful for several applications. Generally, applications that depend on user location are called location-based services (LBS). Past applications have included environment interaction [ACH+01], reminders [DA00, LFR+06], targeted advertising [ACC09], tour guides [AAH+97, CP04], navigation aids, and social networking (e.g., Foursquare and Gowalla).

Outdoor localization is well solved by GPS, but indoor localization remains a challenge in many cases. A wide variety of solutions have been proposed. The early work of Want et al. [WHFG92] and several subsequent works [War98, ACH+01, PCB00, BLO+05, SD05] require that sensors or beacons be installed throughout the environment. This requirement is onerous, but the resulting localization accuracy and reliability are generally excellent.

More recent work has focused on using existing cellular and/or Wi-Fi base stations as radio beacons for localization. In these systems [HFL+04, YA05, HCL+05], location is determined by observing the signal strength of Wi-Fi base stations that are within range. Several efforts have been made to identify semantically-meaningful "places" rather than using simple room labels or coordinates. Kim et al. [KHGE09] give a recent example; they use the appearance and disappearance of radio base stations to indicate entrance into and departure from a place. Other place identification works cluster a time series of location coordinates (obtained by GPS or radio) to identify places, [KWSB04]. The Wi-Fi localization approach of Teller et al. eliminates the need to perform building surveys to build the location database [PCC+10]. There is even a commercial effort that uses wardriving to build a localization system (called Skyhook).

Haeberlen et al. [HFL+04] reported perhaps the most accurate Wi-Fi-localization results to date (along with Youssef et al. [YA05]). Using a Gaussian distribution to statistically model signal strength observations, they reported 95% accuracy over a set of 510 rooms. However, indoor localization is not a solved problem. Looking more carefully at the results, we note that their environment had a dense Wi-Fi infrastructure: at least 5 base stations were in range at all times. Also, their test and training data were drawn from the same

three-minute survey interval. When considering training and test data from different times of day [HFL+04, Figure 8], 70% accuracy was achieved, with most observed errors falling within 5.5 meters. We believe that acoustic localization can close this remaining accuracy gap.

Although Wi-Fi-localization is useful, many areas of interest are not served by Wi-Fi and other areas are served by only a single long-range base station. In the latter case, trilateration is impossible and fingerprinting gives only a very coarse location. There are also some privacy concerns with radio-based localization. Wi-Fi and cellular radios include digital identifiers (such as the MAC address for Wi-Fi) which allow the infrastructure operator to track users' positions (unless users take special privacy precautions). Acoustic techniques require no such identity broadcasting.

Beyond radio Recently, several localization systems have been built without the use of radio. For example, Constandache et al. use a mobile phone's accelerometer and compass for localization [CCR10]. Woodman and Harle [WH08] used a foot-mounted accelerometer to locate a person within a previously-mapped building. The main disadvantage of these dead-reckoning approaches is that they can never determine the absolute location; only changes to position are captured. Therefore, small errors add up over time to create significant location drift; this was observed in my own experiments. Acoustic techniques could complement motion-based approaches.

Scott et al. present perhaps the first acoustic localization work; they place microphones at key locations and users snap their fingers to alert the system of their presence [SD05]. In Chapter 2 we used an ultrasonic sonar sensing system to silently detect the presence of users at computers.

Azizyan et al. (SurroundSense) use a combination of sensors (the microphone, camera, and accelerometer on a mobile phone) to distinguish between neighboring stores in shopping malls [ACC09]. Their problem is made easier by the fact that neighboring stores typically offer different services and therefore a different ambience (including sound, lighting, and decor). Their acoustic processing is straightforward: it considers only audio sample amplitude distribution as a fingerprint. Hence, they distinguish between stores with different loudness characteristics. They report that audio alone did not give good localization accuracy; however, it did work well as the first stage in their multi-sensor localization method.

Other work has used different features to classify ambient sounds. Lu et al. present a framework called SoundSense for classifying sound events recorded on an iPhone and give two examples of its use [LPL+09]. They have an audio feature extraction pipeline similar to that used in speech recognition. SoundSense is ill-suited to localization because it ignores quiet frames [LPL+09, Section 4.1.2] and its audio features are tailored to transient sounds rather than background sounds. However, if a location can be characterized

symbol	meaning	optimal value	Batphone
R_s	sampling rate	96 kHz	44.1 kHz
n_{spec}	spectral resolution	2048 bins	1024 bins
n_{fp}	ABS size	299 bins	325 bins
t_{spec}	frame size	0.1 s	0.1 s
t_{samp}	sampling time	30 s	10 s
	frequency band	0–7 kHz	0–7 kHz
	window function	Hamming	rectangular*

Figure 4.1: System parameters and the values chosen for the Batphone implementation. Section 4.5 shows the effect of altering these parameters. *Newer versions of Batphone use a Hamming window.

by one or more frequent, recurring, transient sound then we expect that SoundSense could be used to aid localization. Our work focuses on persistent sounds and we deal primarily with quiet environments.

A common goal of related work in the audio signal processing community is to assign one of perhaps a few dozen labels to a recording. Sound classes typically correspond to places such as street, nature, construction, car, restaurant, office, or kitchen. Standard audio features such as the mel-frequency cepstral coefficients [DM80], zero-crossing rate, and spectral centroid are used. The system of Eronen et al. performed only slightly worse than human listeners [EPT$^+$06]. Chu et al. achieved similar performance using the matching pursuit (MP) signal decomposition [CNK09]. Our work differs from the above because its goal is not to assign semantic descriptors to recordings as people would, but rather to match previously-learned, specific location labels to recordings.

4.3 Acoustic Background Spectrum

Our localization scheme is based on the Acoustic Background Spectrum (ABS), a new ambient sound fingerprint. The design of the ABS draws from common acoustic signal processing techniques. Figure 4.2 gives an overview of how an ABS is calculated. All the parameters associated with the technique are listed in Figure 4.1. Our scheme works by first calculating the ABS of the room (Sections 4.3.1 and 4.3.2) and then classifying the room by comparing this ABS with the existing, labeled ABS values in a database (Section 4.3.3). We now describe each step in detail.

4.3.1 Spectrogram representation

The first step of our localization technique is recording an audio sample of length t_{samp}. In the next several steps, we transform this signal into a time-frequency representation called a power spectrogram. This involves

1. dividing the recording into frames of length t_{spec},

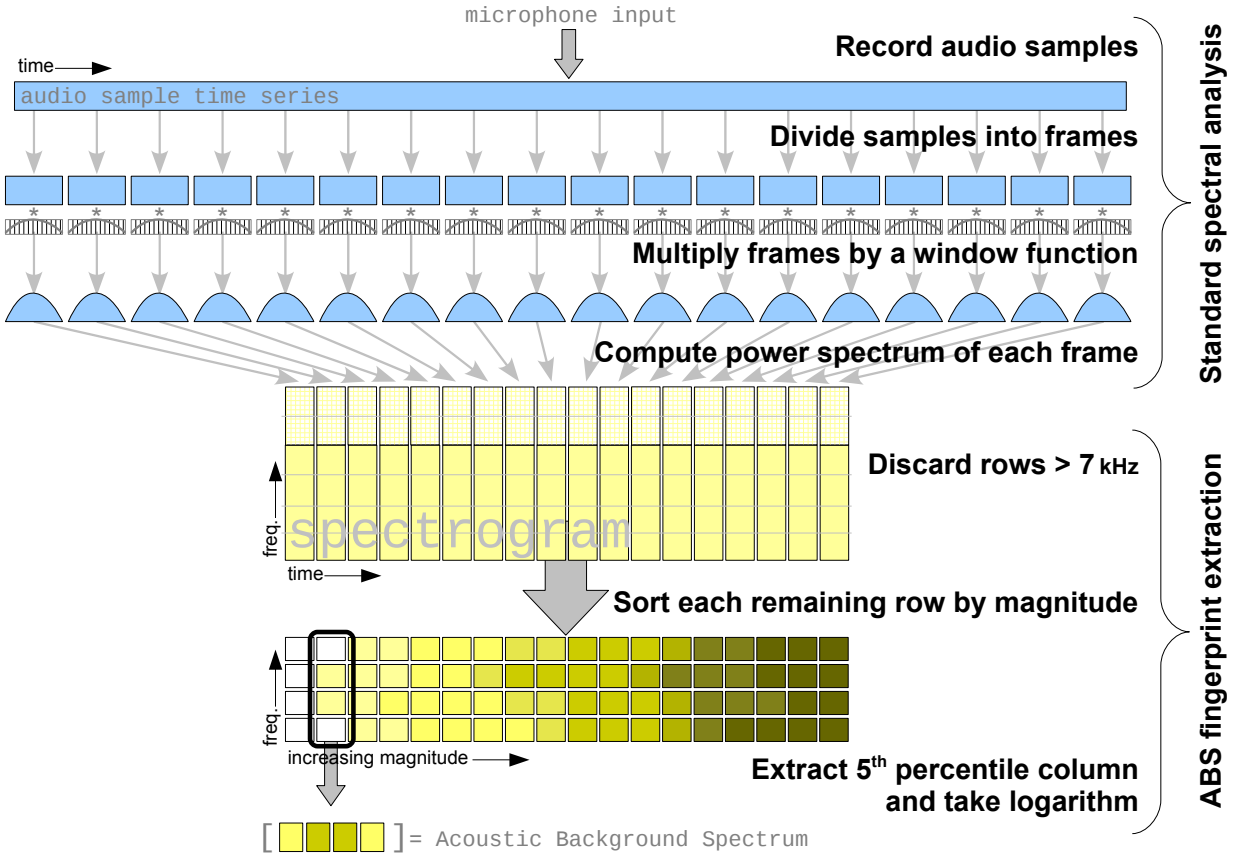


Figure 4.2: Acoustic Background Spectrum fingerprint extraction.

2. multiplying each frame by a window function vector, which reduces the signal magnitudes near the frame boundary, and

3. computing the power spectrum of each frame, which involves

   (a) applying a fast Fourier transform (FFT) of resolution $2 \times (n_{\text{spec}} - 1)$,

   (b) throwing away the redundant second half of the result, leaving n_{spec} elements, and

   (c) multiplying the result elements by their complex conjugates, giving the power.

These are the standard signal processing steps for computing the power spectrogram of a discrete, real-valued signal [OS89, PM96]. Its computation is efficient, with most steps requiring just a scan of the $R_s t_{\text{samp}}$ samples and the majority of time spent computing the $t_{\text{samp}}/t_{\text{spec}}$ FFTs, giving a runtime complexity in the class

$$\Theta\left(\frac{t_{\text{samp}}}{t_{\text{spec}}}(R_s t_{\text{samp}} + n_{\text{spec}} \log n_{\text{spec}})\right).$$

After the spectrogram is calculated, we filter out the frequency band of interest by simply isolating the appropriate rows. The n_{spec} bins of the original spectrogram span the range up to the Nyquist frequency of $R_s/2$. If we are filtering out the 0–7 kHz band, as indicated in Figure 4.1, the number of bins retained is

$$n_{\text{fp}} = \text{round}\left(\frac{7\,\text{kHz}}{R_s/2} n_{\text{spec}}\right). \tag{4.1}$$

The spectrogram quantifies how the frequency content of the recording varies over time. Sounds that are persistent contribute equally to all of the time columns of the spectrogram. On the other hand, relatively short, transient sounds contribute only to a few time columns. We will next describe how to use this difference.

4.3.2 Transient sound rejection

After the spectrogram is computed, we apply a new method for extracting a noise-robust spectrogram summary vector. Intuitively, we would like to filter out transient sounds so that the fingerprint is time-invariant. In other words, we would like to extract the *background* sound levels. As shown in Figure 4.2, we accomplish this by choosing one of the smallest values observed for each frequency during the sampling window. However, the absolute minimum value is sensitive to outliers caused by noise and signal processing artifacts. Instead we choose a value near the minimum, the 5th-percentile value (p05). We expect that transient sounds lasting for less than 95% of the sampling window will be absent from the p05 value since such noises will be manifested as additive features within the upper 95% of the sorted values. Choosing the p05 value involves either sorting or using a linear-time selection algorithm (such as quickselect) on each of the spectrogram rows. Thus, the transient sound rejection step is less computationally complex than the spectrogram calculation step; using linear-time selection, it is in $\Theta(n_{\text{fp}} t_{\text{samp}}/t_{\text{spec}})$. We sort the rows individually since it is more likely that each row will be at least 5% quiet than the full spectrum being quiet 5% of the time. The results in Section 4.5.3 show that the 5th-percentile value gives better performance than the mean, minimum, or other percentile values.

Scaling and normalization The final step in computing the ABS is to compute the logarithm of the spectrogram summary vector. This puts the fingerprint in decibel (dB) units (after multiplying by 10), which is the standard practice in audio signal processing. We validated the benefit of this step via simulation.

Afterward, the fingerprint may optionally be normalized. We do this by dividing the ABS vector by its median value. Ideally, normalization would not be necessary. However, during our experiments, various

hardware gain levels were adjusted to prevent clipping in loud rooms; this made normalization necessary. We expect the median value of the spectrum to remain constant among rooms with different levels of transient sound if and only if those transients are narrow-band. Empirically, this seemed to be the case.

4.3.3 Classification

After the ABS room fingerprint is calculated, it can be compared to previously-observed fingerprints to determine the location. We solve this classification problem via supervised learning. We assume a database of room fingerprints is available where each fingerprint is labeled with a room identifier. The problem at hand is to label the currently-observed room fingerprint. To do this, we choose a distance metric for comparing room fingerprints. In particular, we use the vector Euclidean distance (the city block or Manhattan distance was also evaluated, as shown in Figure 4.8(b)). We then use the simple nearest-neighbor method of classification. This means that we choose the label of the previously-observed room fingerprint with smallest Euclidean distance from the current observation. This can be thought of, equivalently, as choosing the sample with smallest root mean square deviation.

Formally, assuming a set of training pairs (a database)

$$(\vec{fp}_i, roomlabel_i) \in T$$

and a new testing fingerprint $\vec{fp}_{\text{test}}$, the classifier chooses

$$roomlabel_{\text{best}} = \underset{(\vec{fp}_i, \text{roomlabel}_i) \in T}{\text{argmin}} \sqrt{\sum_{j=1}^{n_{\text{fp}}} (fp_i[j] - fp_{\text{test}}[j])^2}. \tag{4.2}$$

Nearest-neighbor classification is required each time a location estimate is desired; the Batphone implementation does this operation every two seconds. Because the fingerprint space has high dimensionality ($n_{\text{fp}} \cong 300$), finding the nearest neighbor can become a bottleneck: we must scan through each fingerprint in the database and compute its distance to the observed fingerprint. This most straightforward approach is clearly not scalable as the database grows to millions of fingerprints. However, nearest-neighbor query latency was imperceptible on Batphone when using a database of a few hundred fingerprints. In future work, we anticipate using a combination of geospatial database queries on a server backend and client-side caching and prefetching to deal with database scaling. We could also use coarse-grained information, such as nearby base station identifiers, to categorize signatures to reduce the search space.

(a) Zoom H4n (b) Apple iPod Touch

Figure 4.3: Experimental platforms. (a) The Zoom H4n handheld recorder was used to collect traces used in the simulations. Its price is ∼$300. (b) The Apple iPod Touch (4th generation) running our Batphone localization software was used for the end-to-end experiment.

It is important to point out that the problem of finding nearest neighbors is common to most fingerprint-based localization approaches. This problem is widely studied and ABS comparison can benefit from advances in the field. As the dimensionality of the fingerprint is reduced, the problem becomes easier. As we describe in Section 4.5.2, ABS fingerprints continue to provide good accuracy even when n_{fp} is reduced to 19.

4.4 Trace collection and simulation

At the onset of this project we speculated that acoustics could be used to identify rooms, but we did not know which acoustic features would be most useful. We used an acoustic-trace-collection approach to gather the data necessary to evaluate acoustic localization. We visited 33 rooms in the Technological Institute and Ford Design Center buildings on the campus of Northwestern University while carrying the recorder shown in Figure 4.3(a), mounted on a microphone stand. The experimenter moved the recorder to four positions in the rooms, each time capturing a 30 second WAV file (24-bit 96 kHz). Stereo recordings were captured, but only the left-channel data was used. Trace collection spanned several weeks, and each room was visited on two different days, giving a total of eight observations per room. We recorded various types of rooms, as shown in Figure 4.4. In our experiments, we did not observe any significant correlation between localization

accuracy and room type or size. We included all the rooms we encountered in those two buildings that were unlocked and which we could access without disrupting class and work activities. Hence, the majority of our samples are from empty, quiet classrooms (Section 4.5.3 deals with noisy rooms). We did not exclude any samples after seeing the results, or for any other reason. Our traces are publicly available on the project web site `http://stevetarzia.com/batphone`.

Collecting these traces allowed us to simulate the performance of acoustic localization while varying several system parameters. In particular, capturing raw, unprocessed recordings gave us the flexibility to study the effects of applying various processing techniques to the same data. The basic simulation approach was to test system performance when observing each of the recordings in turn, assuming that all other recordings had been previously observed; this is called leave-one-out classification.

However, we did not use the full set of 263 remaining samples for training. Specifically, each of the 264 samples (33 rooms $\times$ 2 visits $\times$ 4 positions) was compared to 132 of the samples, which served as the training set. Half of the samples for each room were included in the training set, and the choice of test samples varied according to the room being tested. For the test sample's room, all four samples from the *other* day's visit were included in the training set. For the 32 other rooms, the training set included two samples from each of the two visits. Thus, the training set was unbiased toward any room and it excluded any samples from the same visit as the test sample. As explained in Section 4.5.4, including training data from the same visit as the test data would have made the resulting classification accuracy unrealistically high. Given the training data set described above, the location label for each test sample was assigned from the "closest" training sample; i.e., we used nearest-neighbor classification.

Using this trace collection and simulation scheme, we were able to choose the best parameters for Batphone and to directly compare its performance to that of prior acoustic approaches. These results follow.

4.5 Simulation results

We now describe an evaluation of the ABS technique using the *DECENT* criteria explained in the Introduction. The evaluation uses trace-based simulation and focuses not only on the overall performance of ABS, but also on its sensitivity to its parameters. Our selection of parameter values for the Batphone implementation is based on this study.

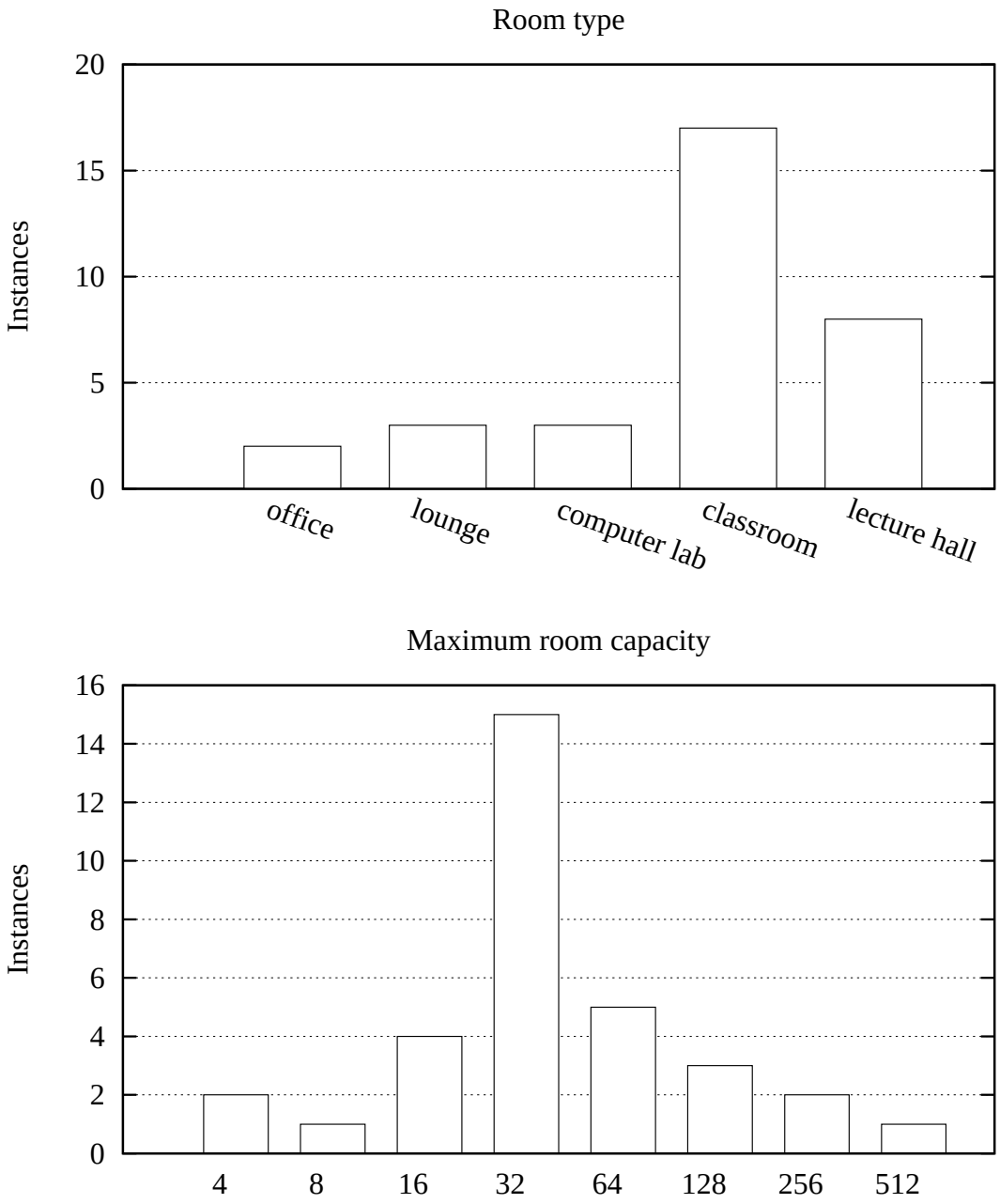


Figure 4.4: Histograms showing the distributions of room type and room size for the traces.

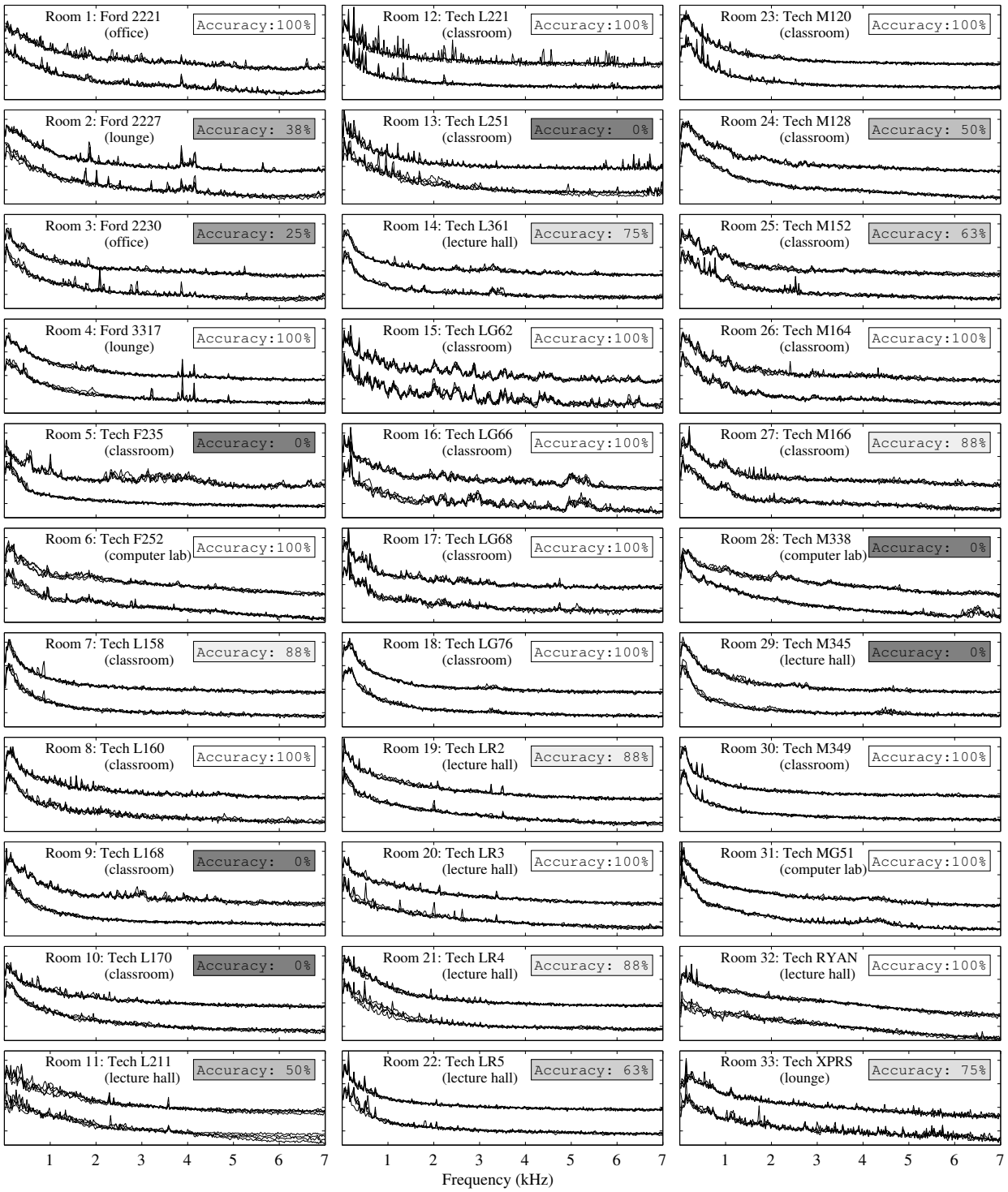


Figure 4.5: Acoustic Background Spectra (ABS) from 33 rooms, using the optimal parameter values. The vertical axis is the ABS value for each frequency bin; these values correspond to the log-scaled power spectrum. Fingerprints from two visits are vertically offset from each other to aid visual comparison. Each visit includes four overlaid fingerprints from different locations within the room.

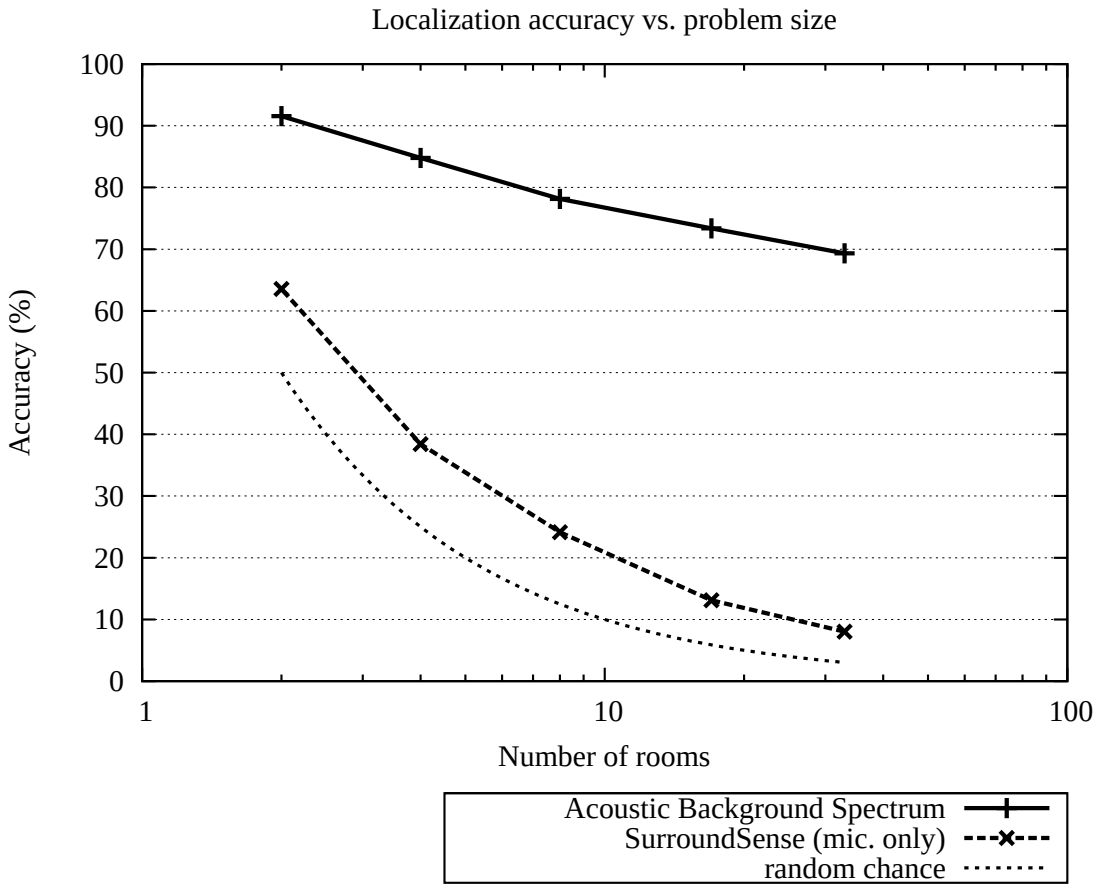


Figure 4.6: Accuracy as a function of the number of rooms being distinguished. 1,000 random subsets of rooms were used to calculate accuracy for problem sizes with fewer than 33 rooms.

4.5.1 Distinctiveness

In order to support localization, a fingerprint should be unique to a single location. In other words, a one-to-one mapping should exist from fingerprints to locations. The role of infrastructure in localization systems is to provide such a mapping. For example, Wi-Fi localization systems rely on the uniqueness of base station MAC addresses. In the absence of uniquely-labeled infrastructure elements, fingerprints are constrained to a finite space and thus overlap of fingerprints is inevitable when the number of locations is sufficiently large. One of the main purposes of our experiments was to evaluate fingerprint distinctiveness, which we measure by the localization accuracy.

Figure 4.6 shows the localization accuracy as a function of the number of rooms being considered, which we call the problem size. For the full set of 33 rooms, localization accuracy is 69%. Pairs of rooms were distinguished with 92% accuracy. Both of these fractions are far higher than the 3% and 50% random

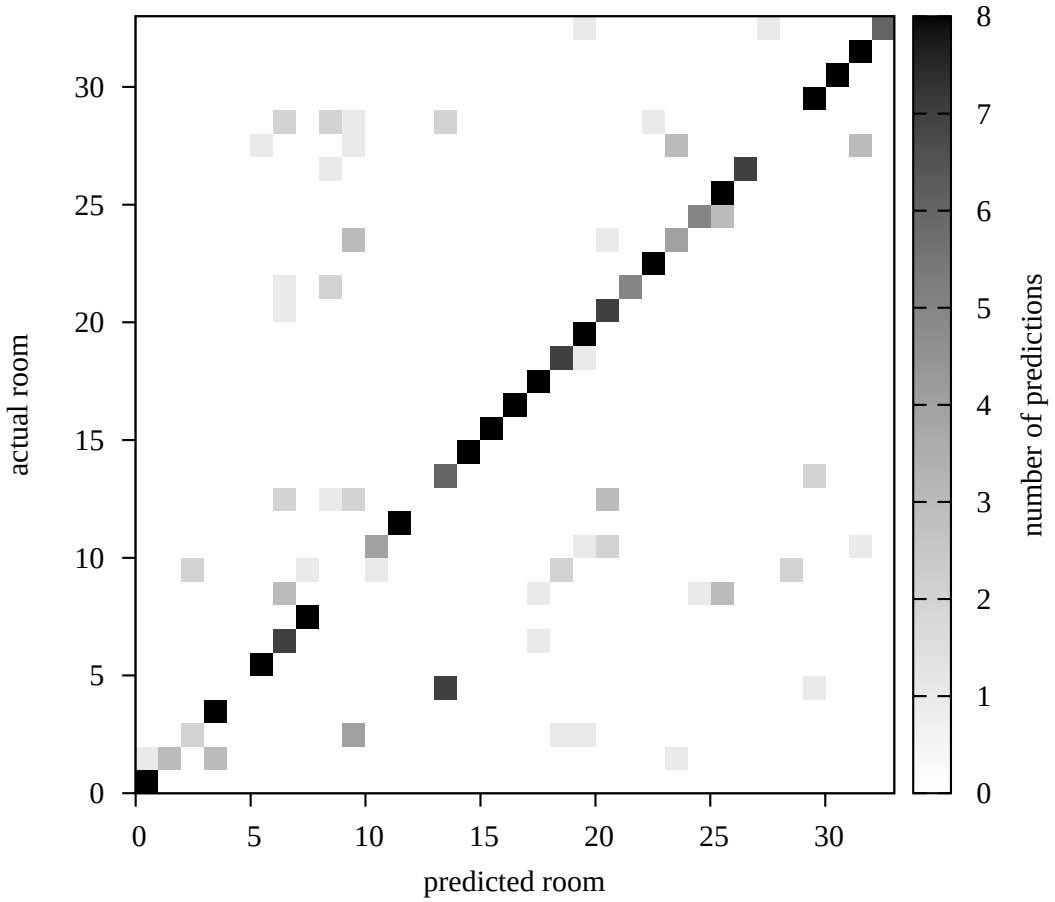


Figure 4.7: Confusion matrix for the 33 room simulation, using the optimal set of parameters. This shows all of the localization decisions made in the simulation. 69% of these location predictions were correct, and thus fall along the diagonal.

chance probabilities. ABS fingerprints also compare favorably to our own implementation of SurroundSense's acoustic fingerprint algorithm [ACC09]. Note that these SurroundSense results include only their acoustic technique, not their full sensor fusion approach (i.e., we do not include the camera and radio).

Figure 4.5 shows the entire set of simulation data for the optimal parameter values; it plots all 264 ABS fingerprints, grouped by room and further grouped into two visits per room. This plot visually confirms the similarity of ABS fingerprints within rooms and distinctiveness between rooms. Figure 4.7 shows the classification confusion matrix for this simulation. We can see that prediction errors, indicated by off-diagonal elements, tend to be distributed across the rooms rather clustered at any particularly-favored predictions.

(a)

Frequency band	Accuracy
full (0–48 kHz)	59.8%
audible (0–20 kHz)	64.8%
low (0–7 kHz)*	69.3%
very low (0–1 kHz)	61.0%
(0–600 Hz)	51.5%
(0–400 Hz)	44.3%
(0–300 Hz)	40.9%
(0–200 Hz)	30.7%
(0–100 Hz)	15.5%
high (7–20 kHz)	28.4%
ultrasonic (20–48 kHz)	25.0%

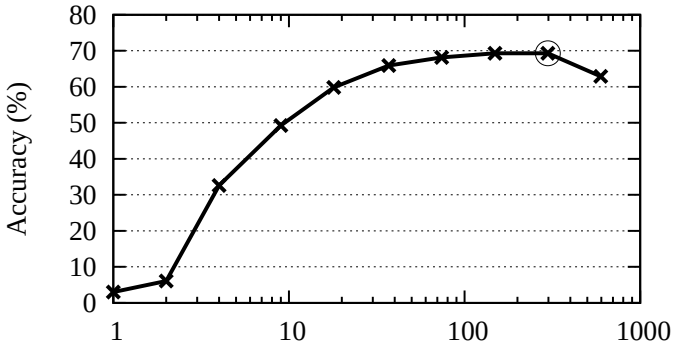
(b)

Distance metric	Accuracy
Euclidean*	69.3%
city block	66.7%

(c)

Spectrogram window	Accuracy
rectangular	65.2%
Hamming*	69.3%
Hann	68.2%
Blackman	67.4%

(d) Fingerprint size (n_{fp}), in floats



(e) Sample time (t_{samp}), in seconds

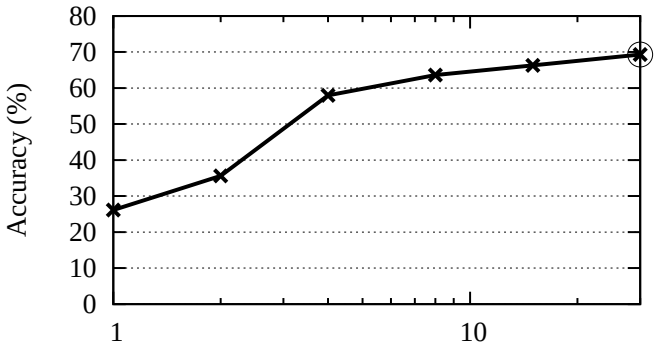


Figure 4.8: Parameter study showing the best localization accuracy achieved for 33 quiet rooms using a variety of signal processing parameters. Optimal values are highlighted.

4.5.2 Parameter study

Figure 4.8 shows how localization accuracy is affected by changes in the ABS fingerprint parameters. Each of these figures represents the change in localization accuracy simulated as one parameter was changed while the other parameters were kept at their optimal values. We kept the problem size constant, using the full set of 264 samples from 33 rooms.

Responsiveness The first parameter we consider is the sample time (t_{samp}), shown in Figure 4.8(e). Intuitively, we expect a longer recording to be more noise-robust and thus to provide better localization accuracy. However, a short sample time is desired for system responsiveness, especially if the user is mobile. Although the best results were obtained using the full 30 second sample time, reducing the sample time to 4 seconds reduced the localization accuracy by only 11 percentage points to 58%. An intermediate value of 8 seconds yielded 64% accuracy. Our sample time is shorter than the one minute sample time recommended in Wi-Fi localization work [HFL+04].

Compactness To reduce memory requirements, a fingerprint should be compact. The ABS fingerprint size can be adjusted by varying the frequency resolution (n_{fp}). However, reducing the fingerprint size generally reduces distinctiveness. Figure 4.8(d) shows the relationship between localization accuracy and the ABS size. Each frequency bin is represented by a floating point value. The optimal size of 299 bins requires 1,196 bytes of storage (assuming single-precision floats). However, if memory is more constrained, ABS fingerprints with only 19 bins (76 bytes) still provide 60% accuracy. Note that accuracy drops when moving to very high resolutions, presumably due to over-fitting.

Frequency band, metric, and window type Figure 4.8(a) shows the effect of limiting the ABS to various frequency ranges. We found that the 0–7 kHz frequency band gave the best results for quiet room samples (in Section 4.5.3 we consider the noisy case). By the Nyquist sampling theorem, this means that any audio sampling rate above 14 kHz would be sufficient to achieve the best results. These requirements are very modest; our iPod supports 44.1 kHz sampling. A sampling rate of at least 8 kHz is required to capture speech. At this low sampling rate, 61% localization accuracy can still be achieved. These results suggest that highly distinctive ABS fingerprints can be captured with even the most basic audio hardware.

The distance metric chosen for comparing fingerprints also affects the results, as shown in Figure 4.8(b). The city block (also called Manhattan) distance requires less than half the arithmetic operations of Euclidean distance, but gives results a few percentage points less accurate. Similarly, as shown in Figure 4.8(c), using

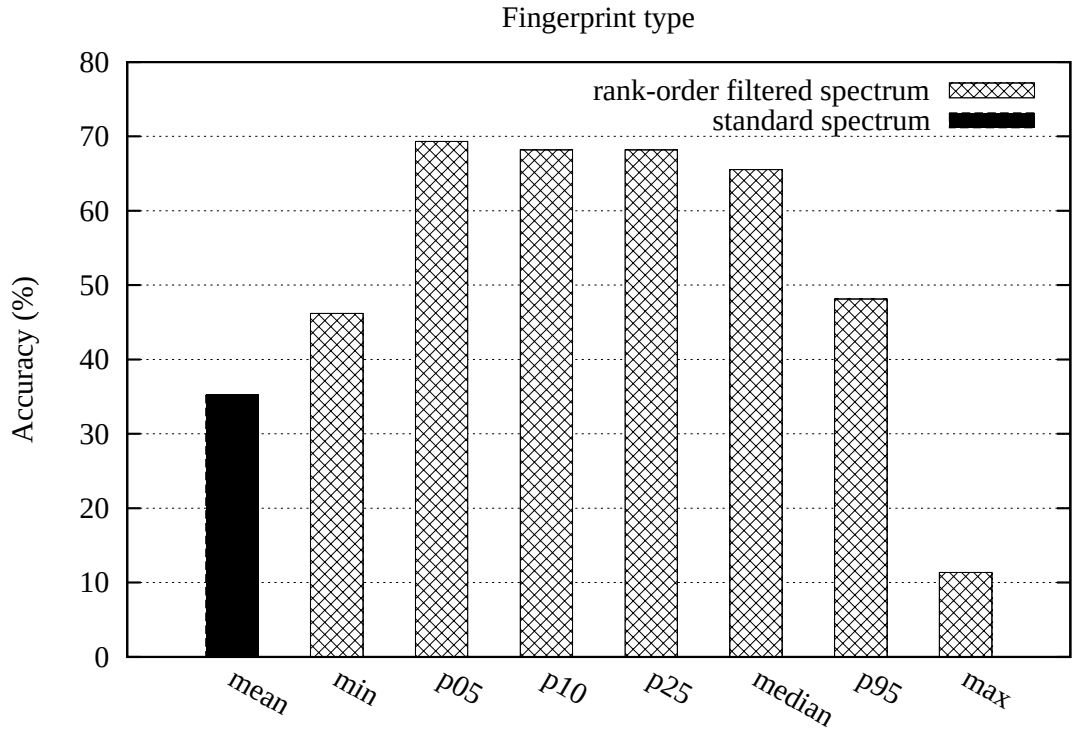


Figure 4.9: The effect of transient sound rejection on localization accuracy. The mean bar represents the using the standard power spectrum as the fingerprint; all other bars represent fingerprints that include rank-order filtering for transient sound rejection, with various values of the rank parameter.

a rectangular window in the spectrogram generation eliminates the vector multiplication cost in exchange for losing a few percentage points of accuracy.

4.5.3 Noise-robustness

Any ambience sensing method for fingerprinting should be robust to temporary changes in the environment, which we call noise. In the design of the ABS, the 5th-percentile (p05) fingerprint extraction step was intended to accomplish this, as described in Section 4.3.2. Figure 4.9 shows how the p05 transient sound rejection method affects localization accuracy. The most important comparison in that bar chart is to the mean. The mean method refers to collapsing the spectrogram into a single column by averaging all of the values across time (the rows) and it is one of the standard methods for computing the power spectrum (Welch's method). The results show that this basic spectral fingerprint produced by the mean achieves 33 percentage points less accuracy than the p05 fingerprint, presumably because it produces fingerprints containing transient sound. Figure 4.9 also shows accuracy achieved using other percentile values, from the

minimum (0th percentile), through the median (50th), to the maximum (100th). The 5th-percentile gave the best results, but other low percentile values have similar benefits compared to the mean.

To test the effect of occupancy noise on the ABS, we made recordings in a lecture hall before, during, and after lectures. This particular lecture hall (Tech LR5) has a maximum seating capacity of 88. Two lectures were recorded: "Economic History of the U.S. to 1865" with 44 attendees and "Real-Time 3D Game Engine Design" with 29 attendees. We defined three occupancy noise states that were observed in the recordings.

- *Quiet* times, prior to lecture, when a few students might be present in the room, but no conversation occurred,
- *Conversation* times, such as during the lecture, when a single voice can be heard, and
- *Chatter* times, directly before and after lecture, when many students were present, talking and moving noisily.

We divided the 162.5 minutes of recordings into 30 second windows and computed the ABS fingerprint of each window. We ran the classification algorithm described in Section 4.3.3 using the same 33 rooms of training data. Thus, the training set included only quiet samples while the test set included quiet, chatter, and conversation samples. This experiment tested the ability of the system to deal with the two types of occupancy noise. We did not expect the occupancy noise to be uniformly distributed across frequency. Therefore, we tested the localization accuracy using a variety of frequency bands in the ABS fingerprint, not just the 0–7 kHz band that gave the best results for the quiet parameter study, as shown in Figure 4.8(a). This allowed us to determine which frequency band is least sensitive to occupancy noise.

Figure 4.10(a) shows the accuracy achieved with the noisy test samples. Accuracy during the chatter noise state was very poor, never significantly passing the random change accuracy of 3.0%. However, the conversation state gave more varied results. Accuracy was poor when using the 0–7 kHz band in the conversation state; p05 transient sound rejection was insufficient. However, 63.4% accuracy was achieved when restricting the ABS to 0–300 Hz. We assume that the success at the 0–300 Hz band is due to its exclusion of the speech band, which is approximately 300–3000 Hz [Moo03]. Unfortunately, as shown in Figure 4.8(a), overall quiet-state localization accuracy dropped from 69% to 41% when omitting the speech band, so there is a trade-off here between quiet-state accuracy and speech-noise rejection. These results suggest that automatically detecting the noise state and switching the ABS band accordingly has the potential to improve the overall success of our approach.

Figure 4.9 showed that p05 transient sound rejection improves over the mean method in the quiet state. The improvement is even more drastic in the conversation state. In that case, using the mean method with

	Occupancy Noise State		
Frequency band	Quiet	Conversation	Chatter
(a) Tech LR5 lecture hall			
full (0–48 kHz)	56.8%	0.0%	0.0%
audible (0–20 kHz)	78.4%	0.0%	0.0%
low (0–7 kHz)	**89.2%**	2.5%	0.0%
very low (0–1 kHz)	45.9%	0.0%	0.0%
(0–600 Hz)	37.8%	7.4%	0.0%
(0–400 Hz)	73.0%	11.4%	0.0%
(0–300 Hz)	75.7%	**63.4%**	0.0%
(0–200 Hz)	21.6%	27.2%	1.2%
(0–100 Hz)	21.6%	26.7%	4.7%
high (7–20 kHz)	21.6	0.0%	0.0%
ultrasonic (20–48 kHz)	32.4%	4.5%	3.5%
(b) Ford 3.317 lounge			
low (0–7 kHz)	**98.2%**	47.2%	—
(0–300 Hz)	87.7%	**79.2%**	—

Figure 4.10: Localization accuracy in two rooms as a function of ABS frequency band and occupancy noise state.

the 0–300 Hz band gave 0% accuracy compared to 63.4% for p05. In summary, a combination of transient sound rejection and band selectivity is needed to deal with speech noise.

We also tested noisy-state accuracy in a smaller room. In particular, we chose a kitchen lounge (Ford 3.317) which is open to a hallway. During an hour-long experiment, we invited graduate students to the lounge to eat free donuts and socialize. A few dozen people came and left. At any time, between 0 and 7 people were present in the room and conversing while recordings were made and the occupancy and noise level was monitored. Figure 4.10(b) shows that, again, switching to the 0–300 Hz band improved accuracy. Conversation-state accuracy was higher in this room than in the lecture hall; this could be due to the presence of longer pauses in speech while chewing donuts or due to the size of the room.

4.5.4 Time-invariance

In order for a room fingerprint to be useful for localization, it must remain relatively constant over time. Our traces spanned several weeks and our simulations excluded same-visit training data. Thus, our results already have factored in the effect that time has on acoustic fingerprints. The high accuracy numbers we report support the claim that ABS fingerprints are stable enough to be used for localization.

Nonetheless, ABS fingerprints did vary somewhat over time. Fingerprint pairs observed on different visits to the same room had signal distance, on average, 24% larger than pairs observed during the same

HVAC state			
Training	Test	Training type	Accuracy
on	on	different visit	71.7%
on/off	off/on	different visit	17.9%
on	on	same visit	98.9%
off	off	same visit	88.0%

Figure 4.11: Localization accuracy for 23 rooms for combinations of HVAC states.

visit. In other words, while fingerprints observed in the same room on different days are similar enough to support accurate localization, fingerprints observed in quick succession are even more similar. In fact, if our simulations had used same-visit training data they would have reported 95.8% localization accuracy. However, that would have been unrealistic since real localization systems must operate using only past training data. The stability characteristics of ABS fingerprints are similar to those of Wi-Fi fingerprints, as reported by Haeberlen et al. [HFL+04]. In that work, localization accuracy dropped from 95% to 70% when same-visit training data was excluded.

HVAC effects Despite the empirical stability of ABS fingerprints, we did identify a scenario in which the ambient sound changed drastically with catastrophic results. Over one weekend, the climate control (HVAC) system in the Technological Institute was shut off for maintenance. We used this rare opportunity to measure the effect that these vents have on the ABS. The results are summarized in Figure 4.11. When using training data from the a different HVAC on/off state, localization accuracy dropped from 71.7% to 17.9% for the 23 rooms accessed over that maintenance weekend. So, changes in HVAC state have a large impact on ABS fingerprints. Yet, as we have already observed, ABS fingerprints remained stable during the main trace collection period. We conclude that HVAC did not significantly change state during trace collection. Of course, some buildings may experience HVAC state changes, and seasonal changes are also possible. However, since such combinations are not numerous, fingerprints representing each of the possible HVAC states can be collected for each room.

At first, we thought that we might identify HVAC sounds as the main source of distinctiveness in ABS fingerprints. However, the samples from the HVAC-off state still exhibited enough distinctiveness to classify with 88.0% accuracy compared to 98.9% accuracy for the HVAC-on state. Note that, because we had only one day to survey the HVAC-off state, we used same-visit training data in these results. These numbers would be lower if testing and training were done on different days, so they should not be directly compared with our other results. Still, they show that the HVAC sounds improve ABS-localization but are not essential.

4.6 Batphone implementation

After completing the trace-based simulation we implemented ABS fingerprinting on Apple's iOS mobile platform. This application, named Batphone, is publicly available on Apple's app store and its GUI is shown in Figure 4.3(b). Batphone allows researchers to perform ABS localization in real-time as well as to capture and export fingerprints for off-line analysis. Our purpose was to evaluate the limitations imposed by a real device's audio hardware and to ensure that the system could operate in real-time.

Batphone is named after the small flying mammals that navigate and locate prey by emitting chirps and then listening to resulting echoes. Batphone, on the other hand, does not emit any sound (nor do our phones fly or eat insects); it follows the same fingerprint extraction steps described in Section 4.3, with some adaptations motivated by implementation considerations.

Hardware limitations Compared to the hardware used in trace collection, our deployed mobile platform has inferior recording capabilities. Its microphone is small, inexpensive, and crowded by other electronics. Also, the audio circuitry is designed for recording speech, and thus has a limited frequency ranges. In particular, audio sampling is 16-bit at 44.1 kHz. The system parameter values chosen were based on the results previously presented and are listed in Figure 4.1.

In a real system, location must be calculated in real-time and on a battery-powered device with limited computation power. We made a few adjustments to the ABS extraction algorithm to reduce its computational load while only slightly reducing its accuracy. In particular, we use a rectangular window function rather than a Hamming window[2] and omit the normalization step. Section 4.8.3 shows our overhead measurements.

On-line training Another complication in the real system is online training and bootstrapping. Recent localization work has explored strategies for building fingerprint databases on-line rather than relying on explicit surveys [PCC+10]. Whenever a fingerprint is observed, the system either matches it with one already observed in the database or adds it to the database as an example of a new room. On-line systems also can automatically adapt to changes in the environment. We do not address the challenges of on-line database growth in this work. Instead, we use the same surveying and simulated training/testing approach that we used in the previous experiment.

Sliding-window 5th-percentile To improve responsiveness in the Batphone implementation, we implemented the ABS calculation steps of Figure 4.2 in a streaming fashion. The spectrogram rows are stored in

[2]Newer versions of Batphone use a Hamming window.

sliding window buffers. After each recording frame arrives, its power spectrum is computed, each frequency element is added to the corresponding spectrogram sliding window, and the oldest element of each window is discarded. At this point the fingerprint is recomputed, based on the new 5th-percentile values from each sliding window. Thus, the fingerprint is updated every $t_{\text{spec}} = 0.1$ seconds rather than every $t_{\text{samp}} = 10$ seconds. Of course, if the user moves to a new location, it will be 10 seconds before all data from the old location are purged from the sliding window. However, this is much better than the 20 second worst-case purge time when not using a sliding window.

The most complex part of the streaming implementation is tracking the 5th-percentile value of the sliding windows. This is known as a rank-order filter [AC89]. In a naïve implementation, a selection or sorting algorithm would be run on the spectrogram rows after each new frame arrives. Instead, we use a much more efficient heap-based data structure to track the p05 value. Values below the current p05 value are stored in a max-heap while values above the current p05 value are stored in a min-heap. Additionally, a mapping between heap indices and a FIFO queue is maintained to allow the oldest data value to be quickly located in the appropriate heap. Details are omitted, but we are simply adapting standard algorithms for calculating the running median to use the 5th instead of 50th percentile [AC89, HS95]. The runtime complexity of a frame insert operation is dominated by n_{fp} heap inserts and is in $\Theta\left(n_{\text{fp}} \log\left(t_{\text{samp}}/t_{\text{spec}}\right)\right)$.

4.7 Linear combination distance

As seen previously in Figure 4.6, the performance of acoustic localization declines as the problem size (the number of rooms to choose from) increases. Our study considered scaling empirically up to 33 rooms.[3] Regardless of the asymptotic scaling of ABS, however, ABS fingerprints could be used to great effect on large-scale problems when combined with another fingerprint such as those from Wi-Fi or cellular radios.

One option is to use a multi-step localization scheme [ACC09]. First, radio localization determines the approximate location, then acoustic localization can effectively determine the final location since it simply has to choose from among a small location neighborhood. We found this approach to be both intuitive and relatively effective. However, we propose a new approach that gives better accuracy by considering the two fingerprint spaces simultaneously; we propose a *linear combination* of each type of fingerprint distances.

The linear combination depends on some measurements of the environment characteristics. Each of n fingerprint types gives a distance d_i, which we expect to fall within some match range (min_i, max_i) when

[3] One goal of making the Batphone implementation publicly available is to allow us to expand our scaling study through volunteers.

comparing two fingerprints observed in the same room. Furthermore, we assign each fingerprint type a weighting constant w_i indicating the relative importance or reliability of that fingerprint type. Combining these constants in the natural way yields the linear combination distance formula:

$$d_{\text{combined}}(d_i \dots d_n) = \sum_{i=1}^{n} w_i \frac{d_i - min_i}{max_i - min_i}. \tag{4.3}$$

In our experiments, min_i and max_i are chosen simply as the minimum and maximum observed same-room distances. In a real system, some sort of outlier elimination would be needed (perhaps by using the 5th and 95th-percentile same-room distances). The weighting constants w_i were chosen experimentally; since only two fingerprint types were considered, this meant just setting $w_1 = 1$ while varying w_2 and choosing the value that resulted in highest localization accuracy.

4.8 Batphone results

Our second experiment was carried out with our Batphone application running on an Apple iPod Touch mobile device (Figure 4.3(b)). The procedure was identical to that described in Section 4.4, except that instead of capturing sound recordings the Batphone app was used to simultaneously capture two compact fingerprints: an ABS fingerprint and a Wi-Fi location coordinate, described below. It involved a set of 43 rooms mostly overlapping with the 33 rooms used in the previous experiments. Also, only two room positions were captured rather than four, meaning that the system had less training data. The two purposes of this experiment were to show the feasibility and performance of a real implementation on typical smartphone hardware and to compare Batphone's localization results to those of the device's Wi-Fi-localization service, which represents the commercial state of the art. We also evaluate the linear combination localization approach.

On our iOS version 4 mobile device, Apple's Core Location service maps Wi-Fi RSSI scans to GPS coordinates. On older versions of iOS, Wi-Fi localization was provided by the Skyhook service. If a cellular or GPS signal is also available, then that information is used as well (however, our test iPod had no such radios). We use the returned (latitude, longitude) coordinate as the Wi-Fi fingerprint, since we do not have access to Apple's internal Wi-Fi fingerprint representation. We compute distances between Wi-Fi fingerprints as simply the traveling distance between coordinates (known as the great circle distance). Core Location provides a dynamic accuracy estimate, which is typically in the tens of meters when using Wi-Fi. This accuracy is far inferior to that achieved by the latest research methods; it is clearly not intended for room-

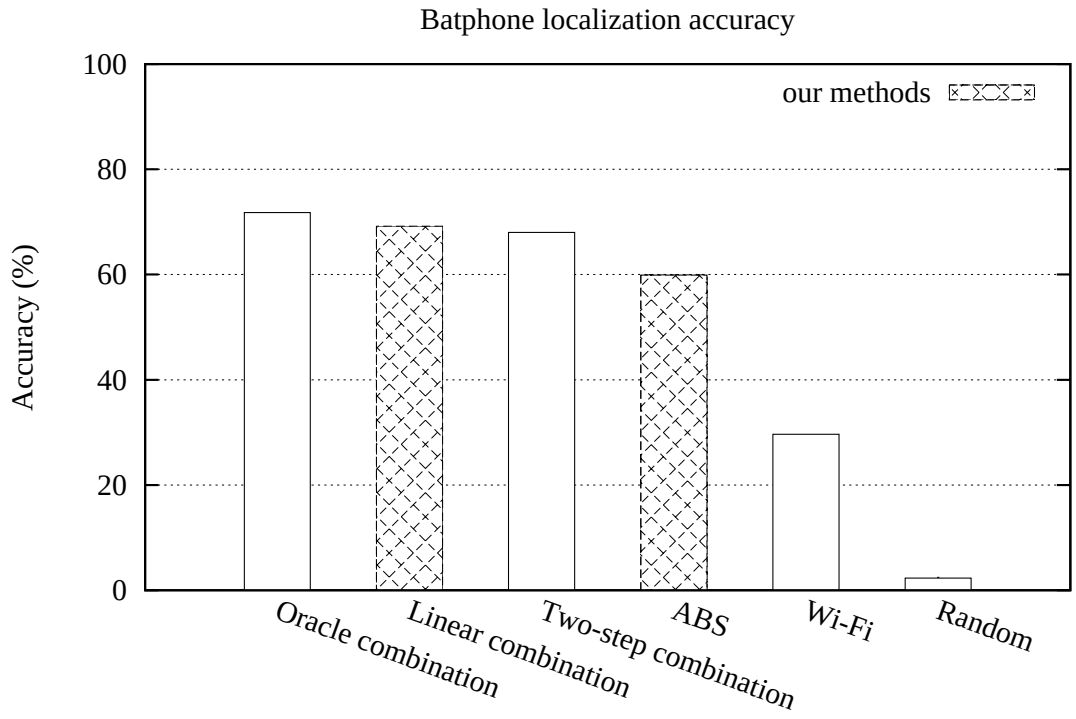


Figure 4.12: Localization accuracy for a set of 43 rooms using a variety of fingerprint types.

level indoor localization. However, Core Location serves as a useful comparison point because it is readily available and well-trained; its database has drawn data from millions of GPS-equipped iPhones. We treat it as a proxy for any of various research Wi-Fi-localization methods operating in sub-optimal conditions, e.g., due to sparse infrastructure, incomplete surveying, or radio interference. We expect such conditions to be quite common. The absolute performance we report for Wi-Fi is far less important than the fact that combining it with ABS yields a very accurate and scalable localization system.

4.8.1 Accuracy

Figure 4.12 shows our Batphone localization results. Wi-Fi performs poorly at room-level localization, as expected. Our Batphone ABS-localization implementation running on real mobile device hardware performs on par with the simulation results summarized in Figure 4.6. It appears that moving from expensive music-recording equipment to a mobile device's built-in microphone had no significant impact on performance. Also, the linear combination distance performed much better than either ABS or Wi-Fi alone. We also compared to a two-step localization approach, which first eliminated all rooms with Wi-Fi distance greater than 35 meters (this value was optimized empirically) and then chose the room with closest ABS distance. This

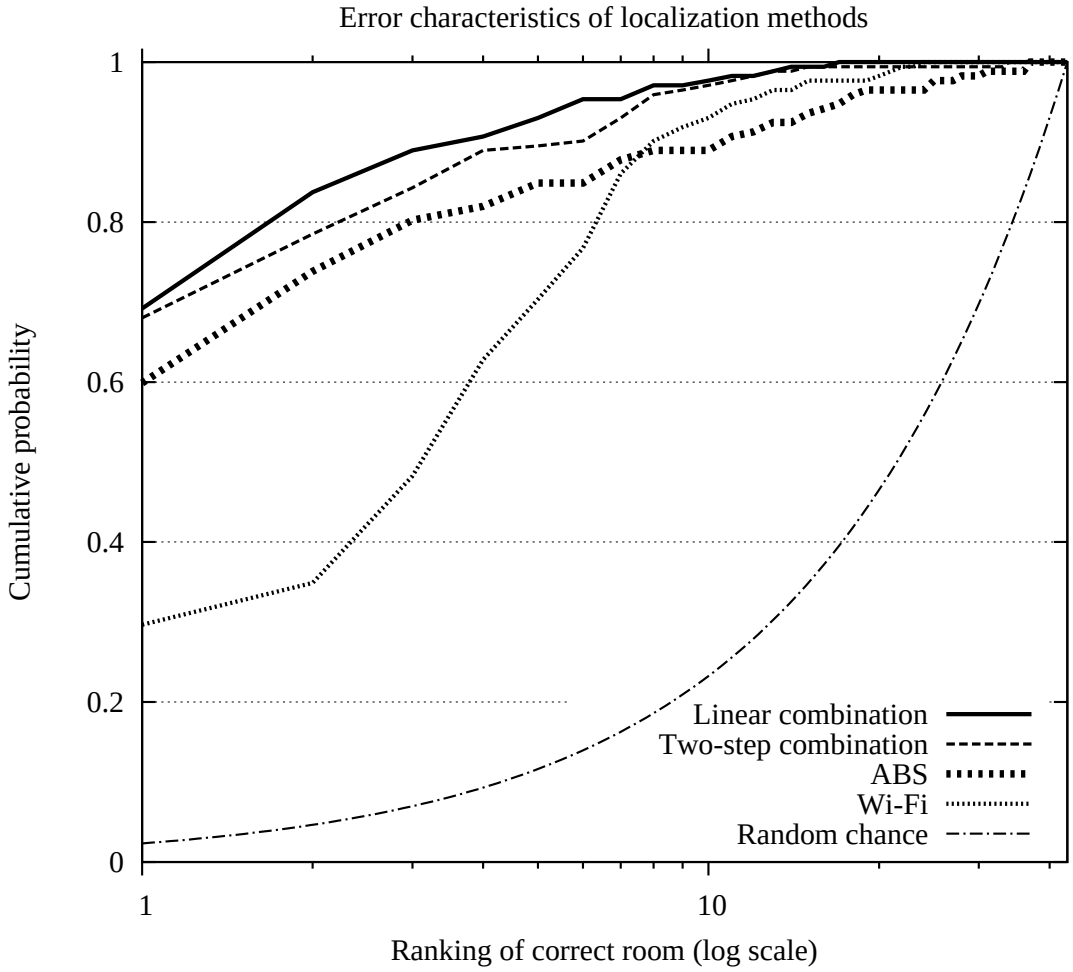


Figure 4.13: Comparison of localization performance for a variety of fingerprint types. Note that Batphone's ABS fingerprint is superior to Wi-Fi's at fine granularity but inferior at coarse granularity. The linear combination approach combines the best aspects of both localization methods.

combination also performed well, but had accuracy 1.5 percentage points inferior to the linear combination. Note that empirically-optimized weighting constants used for the linear combination were $w_{\text{abs}} = 3$ and $w_{\text{wifi}} = 1$.

The oracle combination bar in Figure 4.12 gives a sense of the upper-bound performance of any fingerprint combination scheme. In this hypothetical approach, the algorithm first generates a location estimate using one fingerprint type. Next, an oracle tells whether that location choice was correct, and, if not, the second fingerprint type is used to make a new choice.

The error characteristics were also quite different for the localization methods. Figure 4.13 shows the distribution of rankings for the correct room. These rankings are determined by the sorting the distances

to the closest fingerprint from each room. If the correct room is ranked number 1, then that room is selected as the current location and the localization method is said to be accurate; if the correct room is ranked number 2 then localization was "almost correct", etc. Figure 4.13 shows that acoustic and Wi-Fi localization methods have different strengths. Looking at the left-side of the plot, we see that ABS is much more often strictly-accurate (due, in part, to the relatively small problem size of 43 rooms). At the right-side of the plot, performance of Wi-Fi surpasses ABS; Wi-Fi makes fewer "really bad" errors. The combined localization approaches work far better than either ABS or Wi-Fi, with the linear combination showing a slight advantage over the two-step combination.

4.8.2 Adjacent room discrimination

Unlike radio, sound waves do not penetrate walls well. Therefore, we expect that among adjacent rooms, ABS fingerprints will vary much more significantly than radio fingerprints. To test this hypothesis, we measured the physical distances between the rooms in our study using pixel locations on a floorplan image. Figure 4.14 shows the relationship between fingerprint distances and physical distance. These results are similar to those reported by Park et al. [PCC+10, Figure 2]. Figure 4.14(a) shows, as expected, that Wi-Fi distance tends to increase with physical distance. In particular, no distant room pairs had low Wi-Fi distance; i.e., Wi-Fi has high coarse-grained accuracy. In contrast, Figure 4.14(b) shows that ABS distances of rooms are generally uncorrelated with their physical distances. In particular, nearby rooms have a high degree of diversity (with pairwise distances relatively evenly spread across the ABS distance range). We exploited these complementary features in the linear combination of these two fingerprints, described previously in Section 4.7 and illustrated in the results of Figures 4.12 and 4.13.

Assuming no correlation between ABS and physical distances (as suggested above), Figure 4.6 tells us that pairs of adjacent rooms should be distinguished by ABS fingerprints with around 92% accuracy.

4.8.3 Overhead

To test the overhead associated with ABS fingerprint extraction, we measured iPod battery lifetime while continuously running the Batphone app (with the screen on its dimmest setting). The device ran for 7.9 hours on a single charge, computing about 260,000 fingerprints and making 13,000 location estimates from a database of 190 stored fingerprints (while also rendering an animated GUI showing the results). Battery lifetime was 8.75 hours when running a stripped-down version of the app that simply captured ABS fingerprints without classifying them. For comparison, Wi-Fi localization gave the same 8.75 hour battery life and

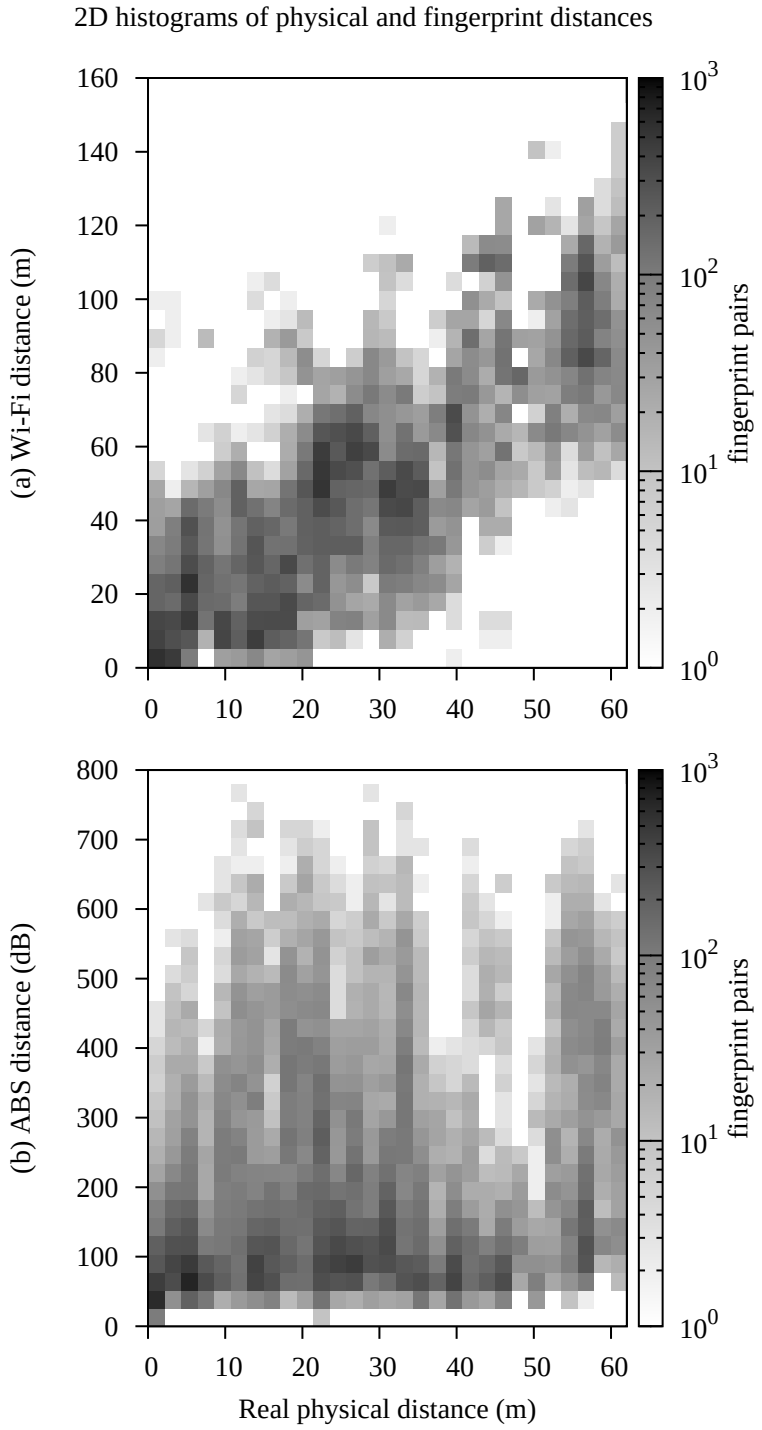


Figure 4.14: Relationship between fingerprint and physical distances.

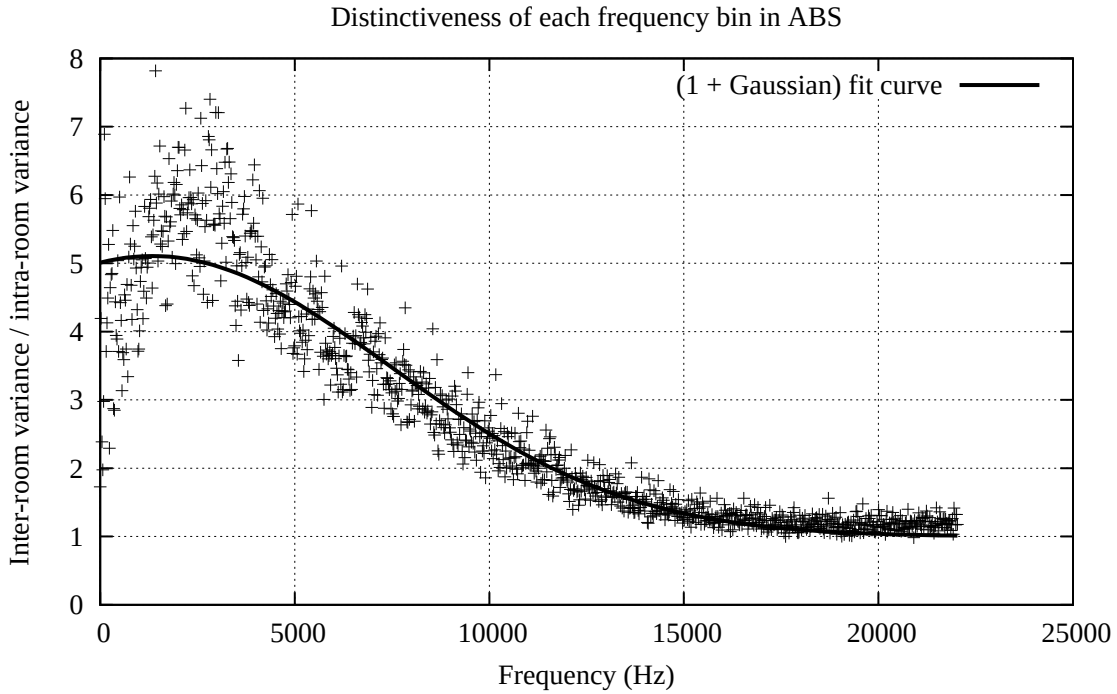


Figure 4.15: The distinctiveness of each frequency bin in the ABS. Gaussian fit with $A = 4.1075$, $\mu = 1343$, and $\sigma = 6112$.

the manufacturer claims up to 7 hours of battery life while playing videos. While continuously determining location and rendering the results at a rate of once per two seconds, Batphone used 11.5% of the CPU and 16.7 MB of RAM (of 256 MB total). These energy and performance overheads certainly seem tolerable.

4.8.4 Dimension study

To determine which frequency bins were most important in the ABS fingerprint we defined a measure of distinctiveness for each dimension; this is simply the average inter-room variance divided by the average intra-room variance. The intuition behind this formula is simple; we want to favor dimensions which show different values for different rooms while being consistent within a room.

Figure 4.15 shows how this measure of distinctiveness varies with frequency; the solid line is an imperfect offset-Gaussian fit. Here we can see that distinctiveness drops off after about 7 kHz which corroborates with the accuracy results in Figure 4.8(a). We tried replacing the band-pass filter with a frequency weighting function based on the fit curve but this did not significantly improve localization results.

4.9 Conclusions

We have shown that accurate indoor localization can be achieved without reliance on a dense, static radio infrastructure. To accomplish this, we introduced a new acoustic ambience fingerprint, the Acoustic Background Spectrum, which is highly distinctive, robust, and easy to compute. We further introduced a method of combining location fingerprints that allows for highly accurate indoor localization by combining two (or more) less accurate fingerprints with different error characteristics.

Future work To fully explore the real-world performance of acoustic localization, we propose building a fingerprint sharing and matching infrastructure that can scale to many users; this remains to be done. Also, we have not fully explored using the linear combination distance with other fingerprint types. It is likely that more accurate localization can be accomplished by adding accelerometer and camera data.

Portions of the noise rejection problem remain unsolved. The 5th-percentile filtering technique does reject some noise and we have a strategy for dealing with single-speaker noise (by switching to the 0–300 Hz band). However, we do not have a policy for controlling this frequency band switch. Furthermore, we were not able to cope with the noise introduced by tens of speaking occupants (the chatter state). We will explore whether adding training samples that include noisy states addresses the latter problem.

In the next chapter, we test the performance of ABS-based localization while walking in hallways and use a history of recent fingerprints to improve accuracy. We also compare to a research-quality Wi-Fi localization system.

Acknowledgments

We would like to thank the following Northwestern University students for helping to collect audio traces for this chapter: Madhav Suresh, Brad Weinberger, Nick Pizzolato, and Jason Lee.

Chapter 5

Markov localization

In this chapter we show how a history of recent acoustic fingerprints, combined with a building floorplan and a simple human movement model, can be used to improve indoor localization accuracy. Our new instance-based Markov localization approach gave 39% accuracy for 170 hall segments in our experiment, compared to 29.5% accuracy for static localization (meaning localization without any history or movement model, as in the last chapter). This experiment also shows that acoustic localization can occur while walking, and in hallways as well as enclosed rooms; thus we can imagine acoustic localization used for indoor navigation. However, acoustic fingerprints from hall segments exhibit more correlation with physical distance, so combining Wi-Fi and acoustic methods did not improve results when in hallways.

5.1 Introduction

In Section 4.7, we saw that Wi-Fi can be used to provide an approximate location which can be further narrowed-down using an acoustic fingerprint. In essence, Wi-Fi reduces the location search space by throwing-out locations that are known to be very distant. Past location information can be used in a similar way to inform the localization algorithm. For example, if you were in Room A ten seconds ago then your current location must be somewhere that is within ten seconds of travel distance from Room A. This Chapter will demonstrate how evidence relating to past location can be used to improve our estimate of the current location.

	sensors			environment			
	mic	Wi-Fi	other	rooms	halls	user movement?	floorplan given?
this chapter	✓	✓			✓	✓	✓
previous chapter	✓	✓		✓			
SurroundSense [ACC09]	✓	✓	✓	✓	∼	✓	
Haeberlen et al. [HFL+04]		✓		✓	✓	✓	✓

Figure 5.1: Summary of differences in problem definitions

5.1.1 Additional inputs

Our problem definition is similar to that in Chapter 4; the goal is to determine the current location given an observed fingerprint and having previously visited all the locations to gather training fingerprints. However, we assume that a few additional inputs are available. Figure 5.1 summarizes the differences between this chapter's problem formulation and that of the last chapter and selected related work.

History In addition to the most recent location fingerprint, the sequence of past fingerprints, taken at regular time intervals is available. This allows us to do probabilistic inference over time, as described later.

Floorplan and movement model A floorplan of the building is available. This allows us to determine which locations are adjacent and thus which location sequences are feasible. If we had lots of user movement traces we could also build a historically-based statistical movement model. Such a model could, for example, give the probability of each possible next location, given the past n locations visited. In this work we have no such movement traces and therefore we simply assume that each possible (adjacent) location is equally likely to succeed a given location.

5.1.2 Related work

The Bayesian inference and Markov movement model described in this chapter have been well-studied for the localization of autonomous robots where it is known as *Markov localization* [Thr02]. However, our problem is somewhat different and more difficult in some ways. In robotics, changes in device position can be well-estimated because the position is controlled by the device itself with outputs like wheel velocities. In our problem the device is instead carried by a person so we have little clue what the direction of movement might be, if any. There is some hope that, in the future, changes in position will be measure-able. Footsteps can actually be detected on fairly reliably using the device's built-in accelerometer [WH08, CCR10] and the absolute direction of movement might be derived from the magnetometer (compass) [QBCN11]. However, accurate dead-reckoning has not yet beet demonstrated with these sensors, and we do not include them in

this chapter.

Haeberlen et al. [HFL+04] present a Wi-Fi localization system which used Markov localization in their mobile user tracking experiments. The performance of Wi-Fi localization suffers when only a few radio scans are observed, and this is the case when the user is walking about. They introduce Markov localization to cope with this drop in absolute accuracy. However, their results do not quantify how much accuracy is gained by adding the Markov movement model. In fact, there appears to be no gain, which is rather confusing. Assuming the same 1.37 m/s walking speed that I observed in my experiments, their accuracy with the Markov model was about 92% (their Figure 10); however, the baseline accuracy is not clearly stated. Their 12,000 square-meter building was divided into 510 localization cells, which means that each cell was about 4.9×4.9 meters. Thus, walking through a cell would take 3.5–5.0 seconds, allowing two or three 1.6 second Wi-Fi scans. Based on this estimate, their accuracy should be about 91–95% for a walking user (from their Figure 3), which is about the same as with the Markov model. This discrepancy could be explained by properties of the walking path used in their simulation. The path may cover location cells that are either smaller than 4.9 meters or have worse-than-average localization accuracy. In any case, the main goal of this chapter is to precisely measure the benefits of Markov localization, apply it to acoustic localization, and describe its derivation and implementation details.

5.2 Markov localization

We use the well-known Markov localization approach but adapt its sensor model for instance-based (nearest-neighbor) classification. This section describes this method in detail.

5.2.1 Bayesian inference

Markov localization relies on Bayesian inference. To do this, we add a history of observations and a floorplan to the problem inputs. In this section we will see how these additions turn the problem into an instance of probabilistic inference over time with a Markov state model. I will refer to observed fingerprints as *evidence* because we no longer determine location based on a single, most recent, fingerprint; instead it is determined with the support of many observations.

Essentially we are adding some state to the problem. We store a *prior* probability distribution: a vector that indicates the likelihood of each possible location hypothesis, based on all the old evidence that we have already seen. Then, when new evidence arrives, we calculate the *posterior* probability distribution using the

Hypotheses	
$x^i_j \equiv S \times T$	the hypothesis that state (location) s_i was observed at time t_j

Inputs	
$T = \langle t_1 \dots t_m \rangle$	the sequence of observation times for the path
$E = \langle e_1 \dots e_m \rangle = e_{1:m}$	the sequence of evidence observations (fingerprints observed on the path)
$S = \{s_1 \dots s_n\}$	the set of possible states (locations)
$P(x^i_t \mid x^j_{t-1}) \in S \times S \to [0,1]$	the transition probability between each pair of states $\langle i, j \rangle$ (derived from floorplan and/or movement traces)
$P(e_j \mid s_i) \in S \times E \to [0,1]$	a sensor model giving the probability of each observations in each of the states (this would be derived from training)

Figure 5.2: Notation for the Markov localization problem

prior, the Markov chain, and the new evidence. The details are now described, following Russell and Norvig [RN03, pg. 542].

To formally define the problem we first specify the inputs and hypotheses, as listed in Figure 5.2. The problem is to deduce the most likely final belief state (location)

$$x^*_m = \underset{x^{i=1\dots n}_m}{\operatorname{argmax}} P(x^i_m \mid e_{1:m}), \tag{5.1}$$

given the inputs. We assume that movement and observations follow a Markov chain. That is, we assume that the next state (location) and observations are drawn from discrete random processes that depend only on the current state.

To solve Equation 5.1, we derive the hypothesis probability formula as

$$P(x^i_t \mid e_{1:t}) = P(x^i_t \mid e_{1:t-1}, e_t)$$

by dividing the observations,

$$= \frac{P(e_t \mid x^i_t, e_{1:t-1}) P(x^i_t \mid e_{1:t-1})}{P(e_t \mid e_{1:t-1})}$$

by Baye's rule,

$$= \frac{P(e_t \mid x^i_t) P(x^i_t \mid e_{1:t-1})}{P(e_t)}$$

by the Markov property of observations,

$$= \alpha_t P(e_t \mid x^i_t) P(x^i_t \mid e_{1:t-1})$$

by defining $\alpha_t \equiv 1/P(e_t)$,

$$= \alpha_t P(e_t|x_t^i) \sum_j P(x_t^i|e_{1:t-1}, x_{t-1}^j) P(x_{t-1}^j|e_{1:t-1})$$

by conditioning on all possible previous states x_{t-1}^j, and

$$= \alpha_t P(e_t|x_t^i) \sum_j P(x_t^i|x_{t-1}^j) P(x_{t-1}^j|e_{1:t-1})$$

by the Markov property of state transitions. The final recursive form of the hypothesis probability is

$$P(x_t^i|e_{1:t}) = \alpha_t \underbrace{P(e_t|x_t^i)}_{\text{sensor model}} \sum_j \underbrace{P(x_t^i|x_{t-1}^j)}_{\text{transition model}} \underbrace{P(x_{t-1}^j|e_{1:t-1})}_{\text{prior}}. \tag{5.2}$$

The factor α_t is a function of the time step and it can be calculated indirectly by constraining the probability sum across hypotheses to equal one:

$$\sum_i P(x_t^i|e_{1:t}) = 1 \tag{5.3}$$

This approach requires that we first calculate the un-normalized probabilities

$$q_t^i \equiv \frac{1}{\alpha_t} P(x_t^i|e_{1:t}) = P(e_t|x_t^i) \sum_j P(x_t^i|x_{t-1}^j) P(x_{t-1}^j|e_{1:t-1}). \tag{5.4}$$

Combining Equations 5.3 and 5.4 gives $\alpha_t \sum_i q_t^i = 1$, so we can calculate

$$\alpha_t = \frac{1}{\sum_i q_t^i} \tag{5.5}$$

and finally

$$P(x_t^i|e_{1:t}) = \alpha_t q_t^i. \tag{5.6}$$

In practice, dynamic programming should be used to efficiently compute all $P(x_t^i|e_{1:t})$ starting at $t = 1$ and proceeding forward in time; this is known as the Viterbi algorithm [RN03]. The initial state probabilities $P(x_0^i)$ could either be given *a priori*, assumed to be uniformly distributed, or taken from the Markov chain's steady state distribution.

5.2.2 Sensor models

The localization formula of Equation 5.2 requires a sensor model, $P(e|x)$, which specifies the probability of every possible observation in every state. In Figure 5.2, we assumed that a sensor model was given as part of the problem inputs. However, in the case of fingerprint-based localization the sensor model is unknown; thus, it is a machine learning problem. We can treat the sensors as "black boxes" and learn a sensor model by recording the sensor outputs for known states. This process is called *sensor model training* and the $\langle x, e \rangle$ pairs recorded are called the *training set*. This section describes the sensor model training process in detail.

One straightforward training approach is to derive an analytical form for the sensor model by choosing a general form and then fitting a set of parameters for each state based on its training set. For example, in the one-dimensional case, we might assume that the observations in each state are drawn from a Gaussian distribution with some characteristic mean and standard deviation that can be learned from the training set.

However, the curve-fitting approach has some obvious drawbacks. Fitting a Gaussian curve with high confidence may require more training points that are available. In our experiments we typically had two to three training samples per location which is far less than what is needed to for a decent fit. Furthermore, we do not generally expect observations to be drawn from a simple distribution such as a Gaussian. Fitting to a general distribution requires an even larger number of training points since every bin of the discretized distribution must be sufficiently sampled. These problems are compounded when a multi-variate Gaussian is needed, as for our high-dimensional fingerprints. We attempted to use a Gaussian sensor model and god very poor static localization accuracy. Reducing the ABS fingerprint dimension from 325 to 21 frequency bins did improve the results, but it was still an order of magnitude worse than the nearest-neighbor approach. So we searched for an alternative.

5.2.3 Instance-based Bayesian inference

For the above reasons, we chose to use *instance-based learning* [RN03, pg. 733]. That is, we use the training data directly in the classification scheme rather than building a sensor model from the training data. The most common such approach is the nearest-neighbor (or k-nearest-neighbors) classification method which we used in Chapter 4 for static localization. A prerequisite for nearest-neighbor classification is to define a distance metric for comparing the similarity of observations. For the Acoustic Background Spectrum we have shown that Euclidean distance works well for location classification.

However, the nearest-neighbor approach has an important shortcoming; it gives an estimate of the most likely hypothesis, but it does not give a measure of confidence for that hypothesis (nor for any of the less-likely

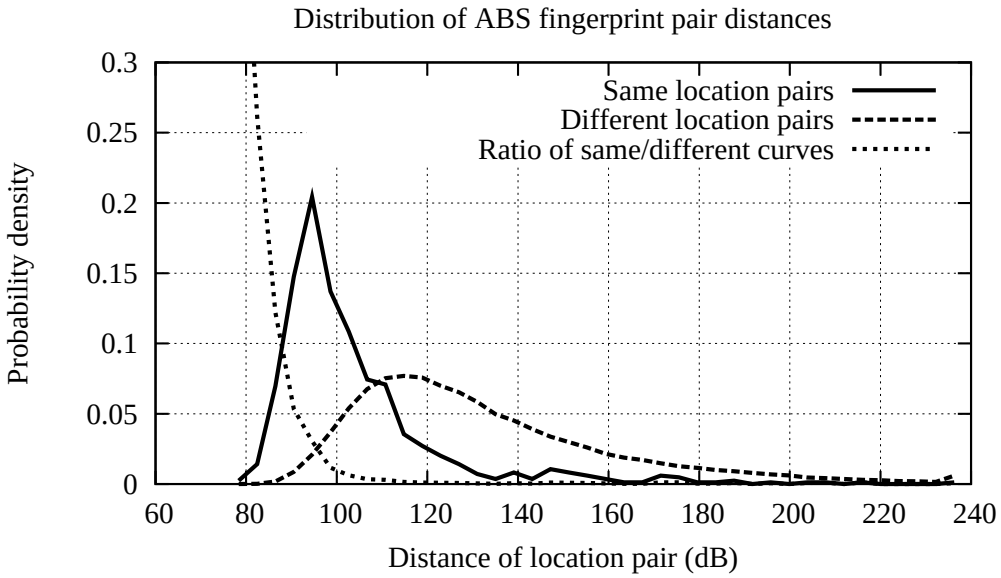


Figure 5.3: Distribution of ABS fingerprint distances within the same location and for different locations. The ratio is normalized so that the probability sum equals one.

hypotheses). In other words, it does not give us the observation probability $P(e|x)$ needed in Equation 5.2 to do Bayesian inference. Of course, the distance is related to the probability and in this section we show how the distribution of observation distances can be used to adapt the nearest-neighbor paradigm to Bayesian inference.

For each candidate state x_i we have a training set

$$\text{Train}_i = \left\{ \epsilon_i^1 \ldots \epsilon_i^{|\text{Train}|} \right\}$$

where each ϵ_i^k is a training fingerprint. For a given observation e the distance for candidate state x_i is defined as that of the nearest training point:

$$\text{dist}(e, x_i) \equiv \min_{\epsilon_i^{k=1\ldots|\text{Train}|}} \text{dist}(e, \epsilon_i^k) \tag{5.7}$$

where the pairwise observation distance can be defined, for example, by the vector Euclidean distance

$$\text{dist}(e_x, e_y) \equiv \sqrt{\sum_i (e_x[i] - e_y[i])^2}. \tag{5.8}$$

Now, our goal is to approximate $P(e|s)$ using the training set directly. We do this by computing the

distance of the evidence to the location's training set $\text{dist}(e_i, s_j)$ and comparing this distance to some expectation of what the distance would be if the evidence was, in fact, from this location. To do this we pre-compute the distribution distances observed for pairs of observations from the same location versus from different locations. The solid line in Figure 5.3 shows $P(\text{dist}(e_i, e_j)|s_i = s_j)$, the distribution of observed distances for pairs of fingerprints from the same location, and the dashed line shows $P(\text{dist}(e_i, e_j)|s_i \neq s_j)$, the distribution of distances for different location pairs.[1] The experiment from which these distances are derived will be described in Section 5.3

We first tried using the distance distribution directly to estimate the sensor model

$$P(e_i|s_j) \approx P(\text{dist}(e_i, s_j)|x_j^i) \tag{5.9}$$

In other words, we used the solid line in Figure 5.3 to map from distance to probability. However, this gave poor results. Intuitively the problem seemed to be that using this same-location curve gave less weight to very small distances than to distances around the peak of 93 dB, because these were rarely observed. On the other hand, the nearest-neighbor approach always favors smaller distances and we wanted to replicate the success we had with nearest-neighbor classification in the last chapter. Our solution was to create a monotonically-decreasing distance probability curve by taking the ratio of the same-location distribution and the different-location distribution, as shown in Figure 5.3. Hence, the sensor model estimate becomes

$$P(e_i|s_j) \approx \frac{P(\text{dist}(e_i, s_j)|x_j^i)}{P(\text{dist}(e_i, s_j)|\neg x_j^i) + \beta} \tag{5.10}$$

where β is a small constant to prevent division by zero (in our case $\beta = 10^{-6}$). In other words, we compute $P(e|s)$ by computing the distance of that observation from the state's closest training fingerprint $\text{dist}(e, s)$ then simply take the value of the dotted curve in Figure 5.3 at that distance. As far as I know, equation 5.10 is novel; it gave much better results than equation 5.9, but its theoretical basis is unclear.

Cucala et al. provide an alternative approach to reconciling the nearest-neighbor method and Bayesian inference [CMRT09].

[1] This experiment is slightly flawed because I am using both training and test data to pre-compute these distance distributions. However, this is a minor point; only the general characteristics of the test set are fed into the algorithm and these characteristics should match those of the training set anyway.

5.2.4 The complete algorithm

Figure 5.4 shows a Matlab implementation of the Markov localization algorithm, using an efficient recurrence (dynamic programming). The algorithm is initialized with a uniform `prior` probability vector. Each time new evidence is observed, `likelihood()` is run to update the location hypotheses. At any given time, the current location estimate is the maximum likelihood state (argmax of the posterior). Note that the inner loop of `likelihood()` can occur in constant time if we use a sparse adjacency-list representation for the Markov chain rather than the simple matrix implementation shown here.

5.3 Experimental setup

We ran an experiment in the Technological Institute building at Northwestern University to test the performance of Markov localization on smartphones using acoustic and Wi-Fi fingerprints. Our test platform was the Apple iPod Touch shown in Figure 4.3(b). We wrote a simple data logging application which recorded the following timestamped data.

- A photograph from the rear-facing camera is snapped every second.
- Audio is recorded and stored in a series of ten second mono WAV files with 44.1 kHz 16-bit sampling.
- A Wi-Fi scan is done every second, recording for each base-station in range its unique MAC address, SSID (network name), and RSSI (received signal strength).

In the experiment I held the device roughly 30 cm in front of my chest with the display facing myself and the camera facing my direction of travel. I planned a walking path through the halls of the building that covered as many hallway segments as possible without re-visiting any; it is a heuristic solution to the Eulerian tour problem. The chosen path was approximately 1.78 kilometers long and is shown in Figure 5.5.

I walked this path twice, on June 9th and 12th, which took 21 to 22 minutes and had an average walking speed of 1.37 m/s. I tried to walk as continuously as possible and at a steady pace. After downloading the logged data, I used the stream of photographs to identify the timestamps at which I reached turning points in the path. These points served as location ground truth and I simply interpolated linearly between turns to get the complete, if approximate, location ground truth. Data from stairways were removed leaving a total of 870 ABS and 2618 Wi-Fi fingerprints.

As shown in Figure 5.5 the floorplan was discretized into $16 \times 16 \times 5 = 1280$ location cells, each measuring 11×11 meters. Of the 1280 total location cells, 170 were covered by the walking path. The goal was to determine in which of 170 hall segments the user was located. Data from the first day was used as the

```
function obs_prob = sensor_model( observation, location, training )
% arguments
%   observation, a fingerprint column-vector of interest
%   location, the index of the location which is being tested, from 1 to N.
%   training, a cell array of all training fingerprints.
%             training{i} is an MxK matrix of K training fingerprints from location i.
% returns
%   obs_prob, the probability of that observation occurring in the location of interest
    K = size(training{location},2); % K is the number of training fingerprints

    % find the distance to the closest training fingerprint for this location, Eqn. 5.7
    for i=1:K
        % calculate the Euclidean distance to a given training fingerprint, Eqn. 5.8
        dist(i) = sqrt( sum( ( observation - training{location}(i) ).^2 ) );
    end
    min_dist = min(dist); % choose the mininum distance

    % Eqn. 5.10, use the empirically-derived dotted curve in Fig. 5.3
    % to map from distance to probability.
    obs_prob = dist_to_prob( min_dist );
end

function posterior = likelihood( evidence, prior, markov_chain, training )
% arguments
%   evidence, the recently-observed fingerprint vector
%   prior, the vector output of the last iteration of this function
%   markov_chain, an NxN sparse matrix where markov_chain(i,j) indicates the probability
%                 of moving from state i to state j.  This is the movement model.
% returns
%   posterior, the probability vector indicating the likelyhood of each state.
%              Argmax of this is the current location estimate.
    N = length(prior);    % N is the number of states
    M = length(evidence); % M is the fingerprint vector length

    for i=1:N % Eqn. 5.4
        % for all possible predecessors
        for j=1:N
            % factor in the transition probability and prior
            posterior(i) = markov_chain(j,i) * prior(i);
        end
        % sensor model factor
        obs_prob = sensor_model( evidence, i, training );
        posterior(i) = posterior(i) * obs_prob;
    end

    % calculate nomalizing alpha, Eqn. 5.5
    alpha = 1 / sum( posterior );
    posterior = alpha * posterior; % Eqn 5.6
end
```

Figure 5.4: Matlab code for Markov localization functions `sensor_model()` and `likelihood()`

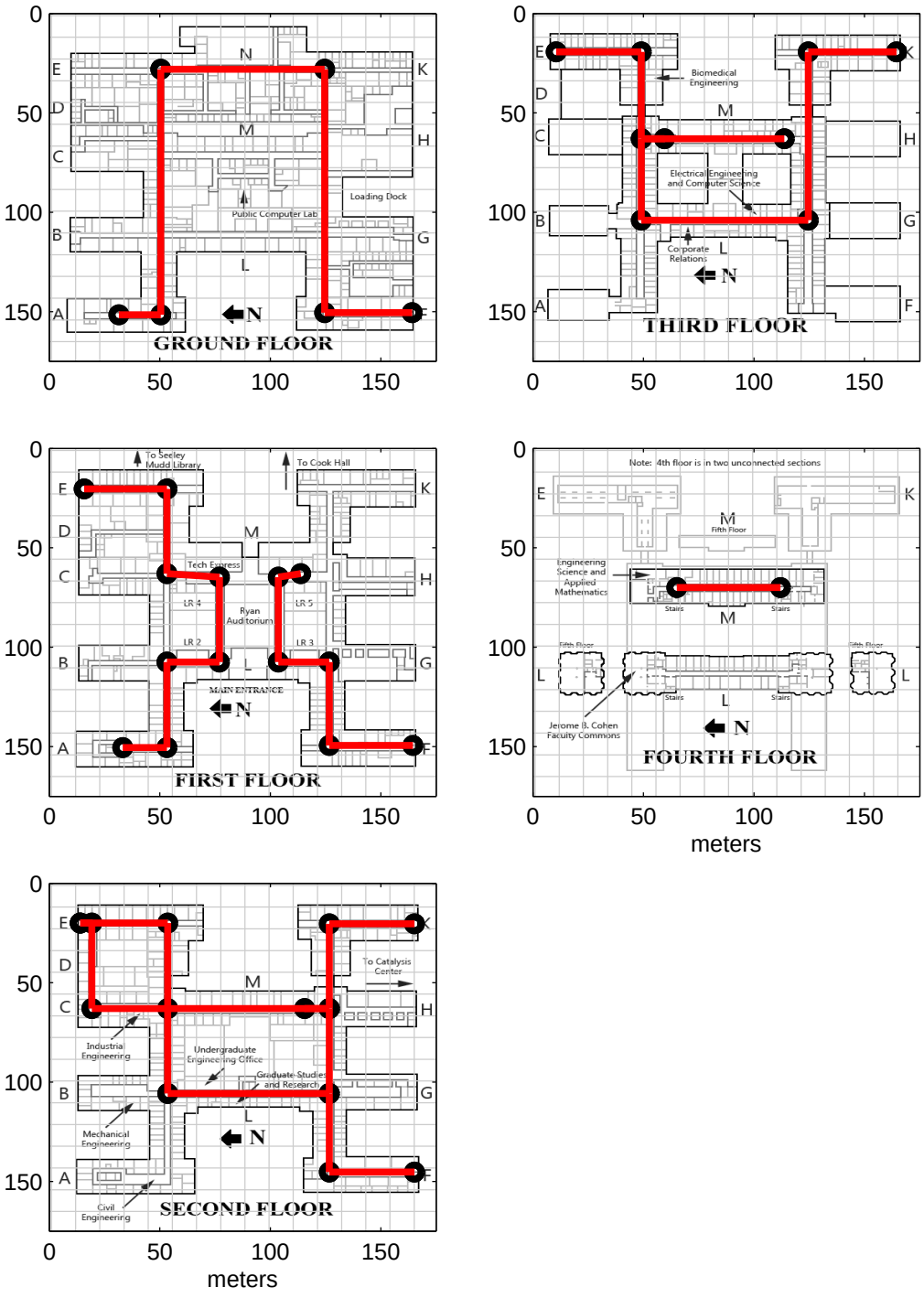


Figure 5.5: Floorplan of the Technological Institute at Northwestern University, the site of the experiments presented in this chapter. The bold line is the experiment's walking path and the grid-lines show how the map was discretized into hall segments. Circles indicate turns and stairways at which the ground truth location was manually recorded from photograph evidence.

training set and the second day was the test set. A Markov movement model was derived from the adjacency relationship of the 170 cells in the experiment. States had varied numbers of out-edges depending on their position in the floorplan; the out-edges of each state were given equal probability. Self-loops were also included in the movement model, allowing the user the stay in the same location for multiple steps. In other words, the Markov movement model was based simply on the constraints of the floorplan with uniform transition probabilities at each state; it generates random feasible walks. By contrast, real walks tend to have some inertia (continued movement in the same direction is very likely) and typically certain hallways in a building are more often walked than others.

For the Markov localization experiments, we generated random, valid state sequences from this Markov chain and assigned to each time-step a random fingerprint observations from the test set. For the static localization experiments, we simply ran localization on each test fingerprint.

5.3.1 Fingerprints

The results in this chapter evaluate Markov localization accuracy for acoustic and Wi-Fi fingerprints. The Acoustic Background Spectrum (ABS), as described in Section 4.3, was used with most of the same parameters used by the Batphone app as shown in Figure 4.1. The only difference is that the sampling time is reduced from ten seconds to three seconds. This allows one or more fingerprint to be captured before the user leaves a location cell.

We used a different Wi-Fi fingerprint than that used in Batphone (and the results of Chapter 4); we use the Wi-Fi fingerprint used by Haeberlen et al. [HFL$^+$04] and others. This fingerprint is a vector of dimension equal to the total number of base-stations in the system where the magnitude of each each dimension is the received signal strength (RSSI) as reported by the Wi-Fi radio scan. If a base-station is out of range (and thus absent from the scan) we assign it a value of -127 which is 28 points lower than the weakest observed signal and 106 points lower than the strongest observed signal. Fingerprint comparisons are then done using the Euclidean distance (Equation 5.8), as for ABS fingerprints.

5.4 Results

Ours results begin with characteristics of the observed fingerprints and are followed by localization accuracy figures.

5.4.1 Fingerprint characteristics

Figure 5.6 shows the correlation between fingerprint distance and the real physical distance for ABS and Wi-Fi fingerprints in this experiment. Comparing to the previous experiment's results, shown in Figure 4.14, we again see a clear correlation for Wi-Fi fingerprints, but there is also some correlation for ABS fingerprints. This is quite different than what we saw in Figure 4.14(b); in that case acoustic fingerprints were not correlated with distance. The important difference here is that we are measuring open hallways rather than sonically-isolated rooms. Thus, we expect sounds to change gradually rather than abruptly when moving around. Unfortunately, this means that in hallways ABS and Wi-Fi are not really orthogonal, and consequently, combining the two did not give improved localization results.

Figure 5.3 shows the distribution of fingerprint distances for pairs of fingerprints from the same hallway segment compared to those from different hallway segments. As expected, pairs of fingerprints from the same location tend to have a smaller distance than pairs from different locations. This result provides some additional evidence to support the use of acoustic localization in indoor hallways. Another important point is that there is significant overlap in the distributions. Therefore, the distance between a test fingerprint and a training fingerprint alone is not a good indication of the confidence of a match. This issue, and our solution of using the ratio curve, was explained in Section 5.2.3.

5.4.2 Markov localization

Figure 5.7 shows the basic Markov localization results when using the ABS fingerprint. In this plot the history length indicates the number of steps since initialization. A history length of one means that no prior is available, so this is the static localization case which only uses one observation; accuracy in this case was 29.5% for the 170 location cells. When the prior becomes well developed, about nine steps into the algorithm, accuracy has improved by 9.5 percentage points to 39%. This difference shows the benefit of Markov localization, and generally of using a history of evidence and a simple movement model. Increasing the history length beyond nine steps did not improve the results.

From Figure 5.7 we can conclude that the Markov localization approach, which makes use of the constraints imposed by the floorplan, provides significant benefits over the static localization method described in Chapter 4. There is some additional computation required for the Markov method, but this is actually very modest; the recursive nature of equation 5.2 allows the hypothesis probability distribution from the previous time step to be re-used as the current time step's prior distribution. Thus, the only real cost associated with the Markov approach is that of obtaining the building floorplan in a suitable machine-readable

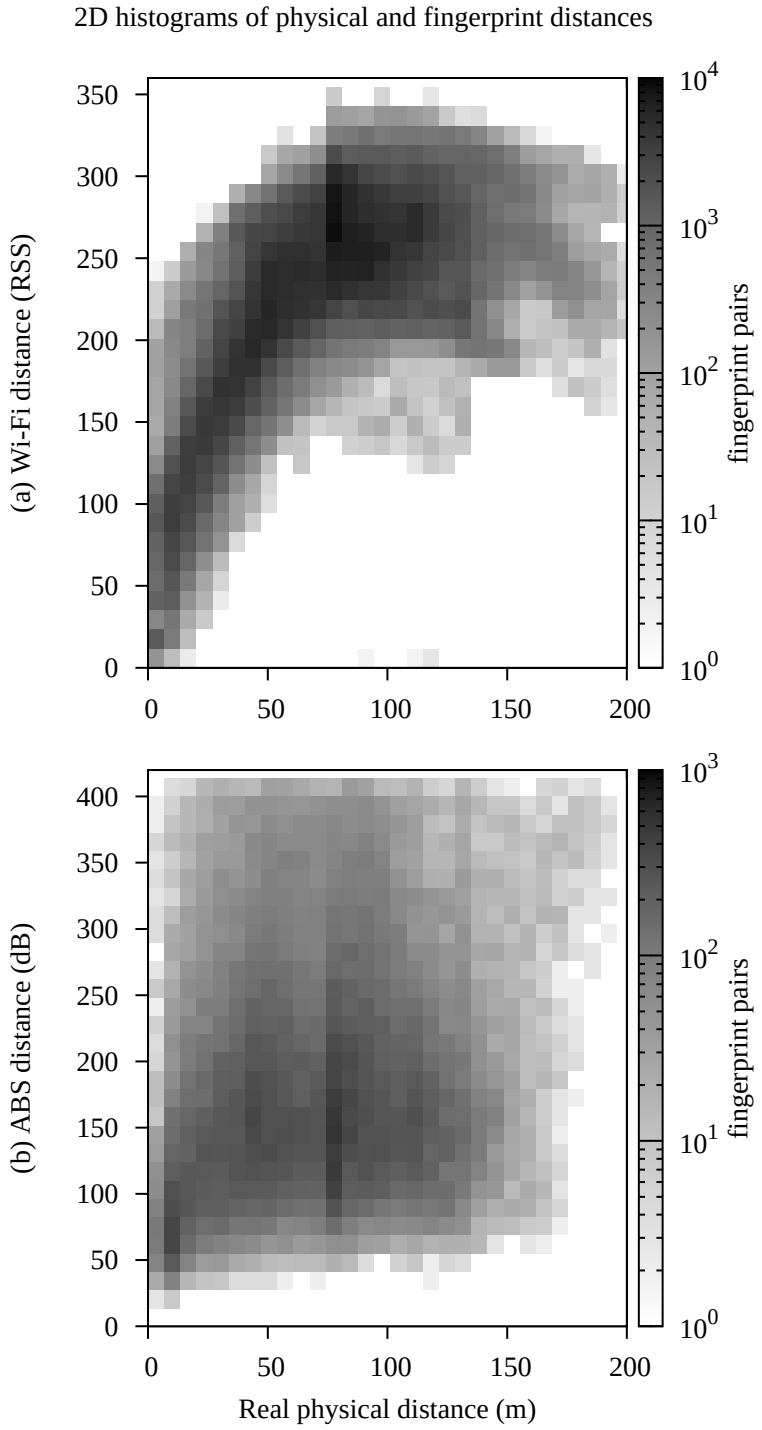


Figure 5.6: Correlation between fingerprint distance and real distance for Wi-Fi and ABS.

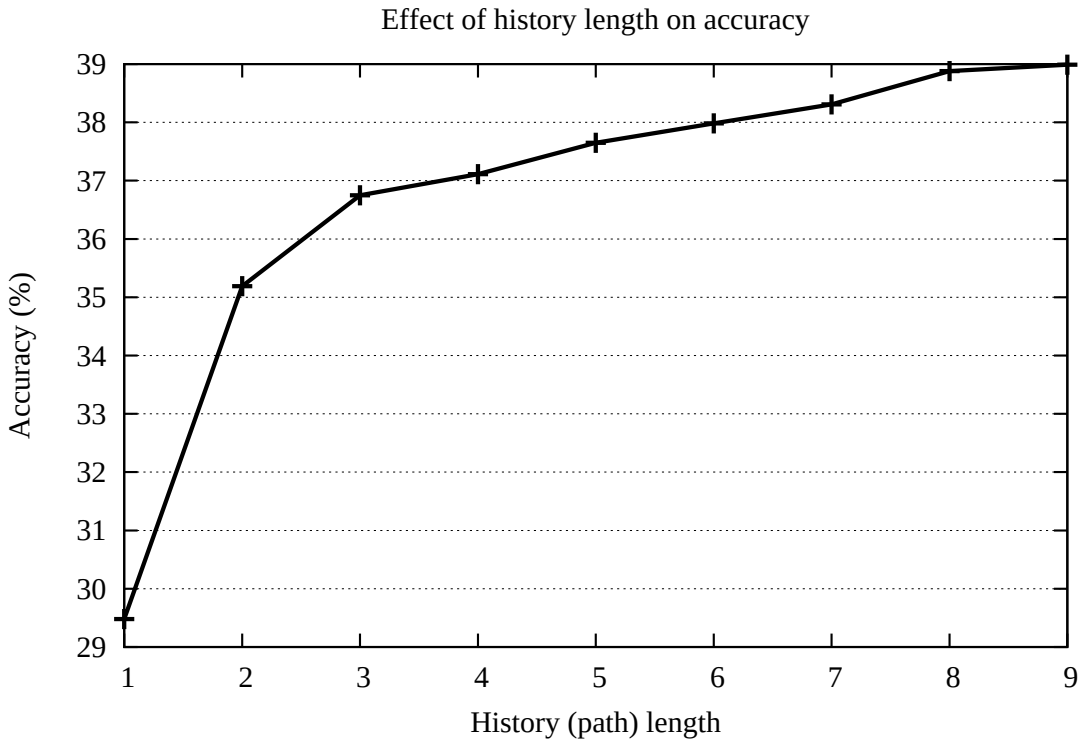


Figure 5.7: Localization accuracy for 170 hall segments using 100,000 randomly-generated path trials. The x-axis represents the number of time steps since beginning the algorithm, or alternatively, the amount of past evidence being used.

format.

5.4.3 Hallway error characteristics

Figure 5.8 shows the distance of the predicted location from the actual location for static localization (without history). The most salient point here is that 98% of Wi-Fi predictions were within 11 meters (one cell) of the correct location. ABS exhibits a less drastic improvement when allowing some error tolerance; 46% of ABS predictions were within 11 meters. This difference can be explained by the stronger correlation of Wi-Fi fingerprint distance with physical distance shown in Figure 5.6. As in Chapter 4, we find that Wi-Fi performs very well at coarse-grain localization.

Figure 5.9 shows another way of looking at the static localization errors; it shows the distribution of likelihood ranks chosen for the correct location. For example, placing the correct location at rank one means that the correct choice was made; rank two means that another location was chosen as most likely, but that the correct location was the second guess, and so on. These results can be compared to those

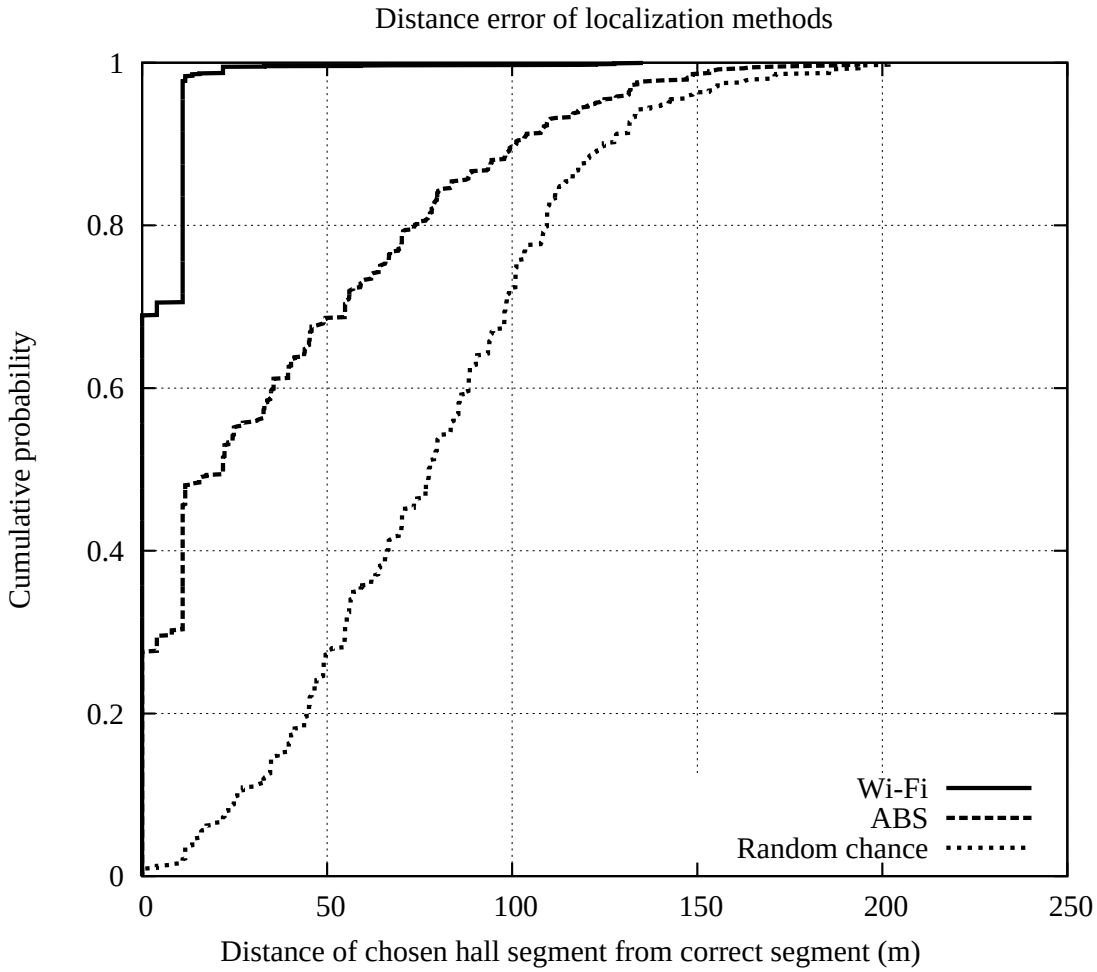


Figure 5.8: Error distance for ABS and Wi-Fi in hallways using static localization.

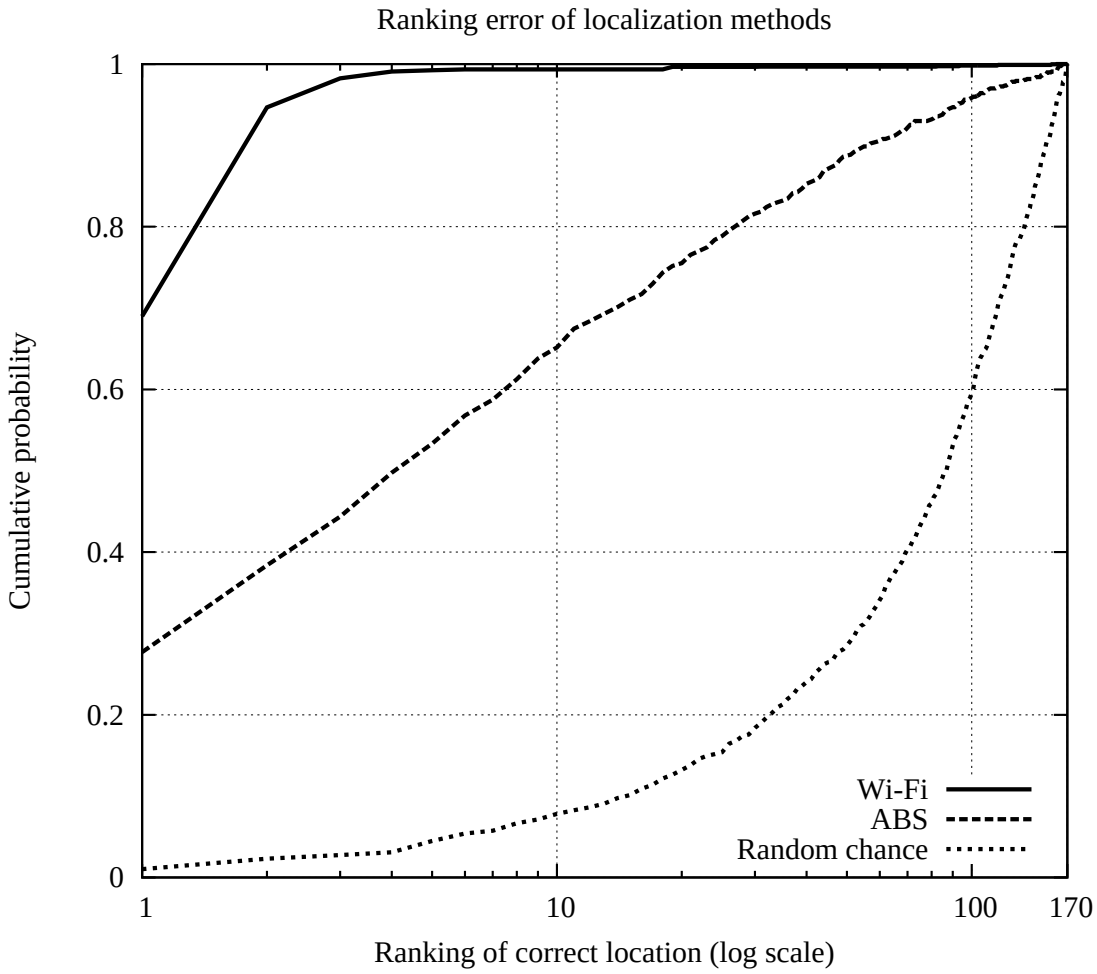


Figure 5.9: Ranking error for ABS and Wi-Fi for 170 hall segments using static localization.

in Figure 4.13 (which was an "easier" experiment because only 33 locations were included instead of 170). Wi-Fi performance is much better here than in Figure 4.13 because we are using a more sophisticated Wi-Fi fingerprint which we train explicitly for the path rather than relying on the database of a commercial service. Looking at the ABS line in Figure 5.9 we again see that allowing some error in the ranking position dramatically improves results. For example, 65% of the time, the correct room was in the "top ten" list of hypotheses.

Figures 5.8 and 5.9 corroborates with prior work [HFL+04, YA05] in showing that Wi-Fi-based localization is very accurate in environments having a dense Wi-Fi base station infrastructure and careful training. It also shows that ABS-based localization can be reasonably accurate in a large building even without any additional sources of location evidence and without any infrastructure.

In the previous chapter, combining Wi-Fi with ABS resulted in improved localization accuracy (see Figures 4.12 and 4.13). This was not case in this chapter's experiment. I tried using both the two-step and linear combination methods with various weighting constants. In all cases, accuracy was inferior to that of Wi-Fi alone. I attribute this both to the improved Wi-Fi fingerprint used in this chapter and the correlation between ABS fingerprint and distance illustrated in Figure 5.6. Whereas in the previous chapter's experiment enclosed rooms were tested and Wi-Fi and ABS fingerprints were found to be orthogonal, in this case open halls were tested and that orthogonality was lost. Acoustic localization is challenging in open halls; we still achieve good accuracy using sound alone, but the synergism with Wi-Fi is lost.

5.5 Conclusions

The above results show that adding user motion to the localization problem, with its accompanying footstep noise and shorter sound sampling, did not degrade ABS localization accuracy. The lower localization accuracies observed relative to the previous chapter can be attributed to the increased size of the problem: the location was chosen from among 170 instead of 43 possibilities. Still, it is surprising and quite promising that we were able to achieve 29.5% localization accuracy in a huge building using only sound.

We also derived an instance-based Markov localization technique for fingerprint-based indoor localization. We showed that by adding a history of evidence and a simple movement mode, Markov localization improved ABS localization accuracy from 29.5% to 39% for 170 locations. Our results also corroborate prior work showing very good indoor localization accuracy for Wi-Fi.

5.5.1 Future work

Our sensor logging application actually captures accelerometer and gyroscope readings are recorded at about 300 Hz, but these data are not yet used. These data might be used to form an estimate of when motion is occurring (and thus, when to run localization) and also what the direction and extent of movement is. Also, an image fingerprint might be extracted from the photographs we have, similar to what was done is SurroundSense [ACC09].

We might also consider the more difficult, but related, problem of simultaneous localization and mapping (SLAM) [Thr02]. In this case, a floorplan would not be provided but learned from the sensor data – particularly the motion data. A more straightforward extension of our work is to use a statistically-based Markov movement model. The idea here would be to effectively reduce the average degree of the model by

giving much more weight to edges joining common pairs of states. For example, it is much more likely that a person walking down a hallway will remain on the hallway rather than enter an adjoining office.

Chapter 6

Conclusions

I started this dissertation by claiming that mobile computers can get a lot more from sound than we might expect, and in particular that "acoustic sensing on mobile computers can provide accurate user presence and location information while requiring no new hardware." This claim is certainly well supported by our results. We are the first to show that active acoustic sensing is possible on laptop computers, and our effective use of ultrasound on these devices is unexpected. That we can distinguish between dozens of rooms by simply listening was also unexpected and we brought a valuable new source of evidence to indoor localization – one of the most important contemporary mobile computing research problems. We found that "quiet" sounds contain useful information, and this has broad implications for acoustic signal processing research and applications. Thus, we have given strong evidence that sound hardware on mobile computers is useful not just for playing alerts and music and for recording speech but also for sensing the environment.

There also have been some immediate practical implications of this work. Our Sonar Power Manager software saves energy on some laptop computers and our Batphone app can determine a smartphone's indoor location, given some training. These two software packages are freely available and they can be easily downloaded by consumers. Both of these software packages require some additional work before deploying them to every laptop computer and smartphone, but these proof-of-concept releases are a good start.

In Chapter 2 we showed that a laptop computer sonar system can reliably detect the presence of a human computer user. However, we found in Chapter 3 that 70% of the surveyed laptop computers were incompatible with the system. Thus, it is not clear whether this system can be widely deployed without custom audio hardware. In Chapter 4 we showed that indoor location can be deduced using only the sound recorded on a smartphone. However, is is unclear for what applications the achieved accuracy (60% for

43 rooms) is sufficient. To deal with the accuracy problem, Chapter 5 showed that a floorplan can be used to improve acoustic localization accuracy somewhat. Wi-Fi scanning is the practical gold standard of localization techniques when indoors; but we showed in Chapter 4 that its accuracy in rooms can be improved by incorporating acoustic evidence. However, in Chapter 5 we found that, in open hallways, there was no benefit to combining acoustic and Wi-Fi evidence. So, it seems that an indoor smartphone localization system should

- rely heavily on Wi-Fi scans if the device has such a radio and when base stations are in range,
- use a building floorplan when provided by the building owner, and
- use ABS fingerprints when Wi-Fi is unavailable or to augment Wi-Fi in closed rooms.

For the latter point, some method of inferring whether the current location is a closed room or an open hallway is needed.

Indeed, this work is only a small start: both for indoor localization and for mobile computer acoustic sensing in general. To get a sense of what kind of physical sensing is possible using sound we can look to nature. In the 1970s, biologists began studying the biosonar used by microchiropteran bats and marine mammals such as dolphins [Sim79, TMV04]. Their abilities are, in fact, much more sophisticated than any man-made sonar system yet developed. Bats in particular use sound to measure the precise location, orientation, and trajectory of their insect prey with astonishing accuracy. Scientists know relatively little about how biosonar works, but their observations tell us that sound paints a detailed picture of the environment. Someday, I hope that we will be able to see in sound with all the brilliant clarity of those unpopular, but astonishing, furry, flying, acoustic-sensing animals.

6.1 Future work

I think that this dissertation has opened the door to quite a lot of future work, some of which was already mentioned in the preceding chapters. The future applications that could be built using presence and location were already mentioned in the introduction, but there is also a lot more that can be done to develop the sensing techniques. In Chapter 2 we show that active acoustic sensing can give user presence, but much more may be possible with this technique. We may be able to correlate the sensed motion level with user engagement, which would be of interest to application and system designers. Simply installing a cheap ultrasound emitter on the computer would certainly improve presence detection reliability while it might also allow more detailed sensing.

This dissertation does not discuss active methods for fingerprinting rooms; theoretically, active sensing does seem quite promising. Our own attempts to characterize the room by its measured impulse response produced results inferior to the (much easier) passive method, but these negative results are certainly not conclusive. It also seems possible to opportunistically use noise sources in the environment, such as footsteps, to measure a room's impulse response.

We also have some preliminary results showing that the ultrasound emissions from fluorescent lights can be used to recognize certain rooms, but this technique has not been fully tested. I made some recordings with another set of audio equipment that was capable of 192 kHz sampling; this allowed us to look at the ultrasound spectrum up to 96 kHz. I found that a fraction of the rooms visited in the experiment of Chapter 4 exhibited one to three very strong, narrow spikes in the energy spectrum in the 30-96 kHz range. Despite this observation, Figure 4.8(a) shows that our ABS fingerprint extraction steps applied to the the ultrasonic range did not give good localization accuracy. Nonetheless, I think that these ultrasound peaks can be somehow used to improve passive acoustic localization accuracy.

There is a lot of work to be done in order to make large scale indoor localization practical. Crowd-sourcing the training process offers tremendous benefits, but it leads to unreliable data which must be somehow sanitized; good incentives are also needed to encourage user participation. There is still not much good theory to support localization with multiple types of sensors such as sound, photo, motion, and radio. Also, automatic map-building techniques would greatly expand the reach of indoor localization; this is called simultaneous localization and mapping (SLAM) in the artificial intelligence community.

I will close with a personal opinion: that acoustic sensing will have an important role to play in our future. It is my sincere hope that this dissertation will inspire work that I myself would never have dreamed of.

Bibliography

[AAH+97] Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton, *Cyberguide: A mobile context-aware tour guide*, Wireless Networks **8** (1997), 421–433.

[AC89] Jaakko T. Astola and T. George Campbell, *On computation of the running median*, IEEE Trans. on Acoustics, Speech and Signal Processing **37** (1989), no. 4, 572–574.

[ACC09] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury, *SurroundSense: mobile phone localization via ambience fingerprinting*, Proc. Intl. Conf. on Mobile Computing and Networking (MobiCom), 2009, pp. 261–272.

[ACH+01] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper, *Implementing a sentient computing system*, IEEE Computer **34** (2001), no. 8, 50–56.

[BB97] Avrim Blum and Carl Burch, *On-line learning and the metrical task system problem*, Proc. Conf. on Computational Learning Theory (COLT), July 1997, pp. 45–53.

[Bea09] Thomas Beauvisage, *Computer usage in daily life*, Proc. Intl. Conf. on Human Factors in Computing Systems (CHI), 2009, pp. 575–584.

[BLO+05] Gaetano Borriello, Alan Liu, Tony Offer, Christopher Palistrant, and Richard Sharp, *WALRUS: wireless acoustic location with room-level resolution using ultrasound*, Proc. Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys), 2005, pp. 191–203.

[CCR10] Ionut Constandache, Romit Roy Choudhury, and Injong Rhee, *Towards mobile phone localization without war-driving*, Proc. Intl. Conf. on Computer Communications (INFOCOMM), 2010.

[CMRT09] Lionel Cucala, Jean-Michel Marin, Christian P. Robert, and D. M. Titterington, *A Bayesian reassessment of nearest-neighbor classification*, J. of the American Statistical Association **104** (2009), no. 485, 263–273.

[CNK09] Selina Chu, Shrikanth Narayanan, and C.-C. Jay Kuo, *Environmental sound recognition with time-frequency audio features*, IEEE Trans. on Audio, Speech and Language Processing **17** (2009), no. 6, 1142–1158.

[CP04] Carmine Ciavarella and Fabio Paternò, *The design of a handheld, location-aware guide for indoor environments*, Personal and Ubiquitous Computing **8** (2004), 82–91.

[DA00] Anind K. Dey and Gregory D. Abowd, *CybreMinder: A context-aware system for supporting reminders*, Proc. Intl. Symposium on Handheld and Ubiquitous Computing (HUC), 2000, pp. 172–186.

[DE03] Alice B. Dalton and Carla S. Ellis, *Sensing user intention and context for energy management*, Proc. Wkshp. on Hot Topics in Operating Systems (HotOS), May 2003.

[Din99] Peter A. Dinda, *The statistical properties of host load*, Scientific Programming **7** (1999), no. 3–4, 211–229.

[DM80] Steven B. Davis and Paul Mermelstein, *Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences*, IEEE Trans. on Acoustics, Speech, and Signal Processing **28** (1980), no. 4, 357–366.

[DMD+07] Peter A. Dinda, Gokhan Memik, Robert P. Dick, Bin Lin, Arindam Mallik, Ashish Gupta, and Samuel Rossoff, *The user in experimental computer systems research*, Proc. Wkshp. on Experimental Computer Science (ExpCS), 2007.

[EN81] David W. Embley and George Nagy, *Behavioral aspects of text editors*, ACM Computing Surveys **13** (1981), no. 1, 33–70.

[EPT+06] Antti J. Eronen, Vesa T. Peltonen, Juha T. Tuomi, Anssi P. Klapuri, Seppo Fagerlund, Timo Sorsa, Gaëtan Lorho, and Jyri Huopaniemi, *Audio-based context recognition*, IEEE Trans. on Audio, Speech and Language Processing **14** (2006), no. 1, 321–329.

[HBD96] Mor Harchol-Balter and Allen B. Downey, *Exploiting process lifetime distributions for dynamic load balancing*, Proc. Intl. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS), May 1996, pp. 13–24.

[HCL+05] Jeffrey Hightower, Sunny Consolvo, Anthony LaMarca, Ian Smith, and Jeff Hughes, *Learning and recognizing the places we go*, Proc. Intl. Conf. on Ubiquitous Computing (UbiComp), August 2005, pp. 159–176.

[HFL+04] Andreas Haeberlen, Eliot Flannery, Andrew M. Ladd, Algis Rudys, Dan S. Wallach, and Lydia E. Kavraki, *Practical robust localization over large-scale 802.11 wireless networks*, Proc. Intl. Conf. on Mobile Computing and Networking (MobiCom), 2004, pp. 70–84.

[HS95] Wolfgang Härdle and William Steiger, *Optimal median smoothing*, Applied Statistics **44** (1995), no. 2, 258–264.

[KHGE09] Donnie H. Kim, Jeffrey Hightower, Ramesh Govindan, and Deborah Estrin, *Discovering semantically meaningful places from pervasive RF-beacons*, Proc. Intl. Conf. on Ubiquitous Computing (UbiComp), 2009, pp. 21–30.

[KKK+11] Jae Min Kim, Minyong Kim, Joonho Kong, Hyong Beom Jang, and Sung Woo Chung, *Do you waste laptop battery to light the room?*, IEEE Computer (2011).

[Kom97] Akinori Komatsubara, *Psychological upper and lower limits of system response time and user's preferance on skill level*, Proc. Intl. Conf. on Human Factors in Computing Systems (CHI), August 1997, pp. 829–832.

[Kut73] Heinrich Kuttruff, *Room acoustics*, Halsted Press, 1973.

[KWSB04] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello, *Extracting places from traces of locations*, Proc. Intl. Wkshp. on Wireless mobile applications and services on WLAN hotspots (WMASH), 2004, pp. 110–118.

[LFR+06] Pamela J. Ludford, Dan Frankowski, Ken Reily, Kurt Wilms, and Loren Terveen, *Because I carry my cell phone anyway: functional location-based reminder applications*, Proc. Intl. Conf. on Human Factors in Computing Systems (CHI), 2006, pp. 889–898.

[LMD+09] Bin Lin, Arindam Mallik, Peter A. Dinda, Gokhan Memik, and Robert P. Dick, *User- and process-driven dynamic voltage and frequency scaling*, Proc. Intl. Symposium on Performance Analysis of Systems and Software (ISPASS), April 2009.

[LPL+09] Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell, *SoundSense: scalable sound sensing for people-centric applications on mobile phones*, Proc. Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys), 2009, pp. 165–178.

[MCD+08] Arindam Mallik, Jack Cosgrove, Robert P. Dick, Gokhan Memik, and Peter A. Dinda, *PICSEL: measuring user-perceived performance to control dynamic frequency scaling*, Proc. Intl. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2008, pp. 70–79.

[ML91] Matt W. Mutka and Miron Livny, *The available capacity of a privately owned workstation environment*, Performance Evaluation **12** (1991), no. 4, 269–284.

[Moo03] Brian C. J. Moore, *An introduction to the psychology of hearing*, Emerald Group Publishing, 2003.

[MSS03] Anil Madhavapeddy, David Scott, and Richard Sharp, *Context-aware computing with sound*, Proc. Intl. Conf. on Ubiquitous Computing (UbiComp), 2003, pp. 315–332.

[MSST05] Anil Madhavapeddy, Richard Sharp, David Scott, and Alastair Tse, *Audio networking: the forgotten wireless technology*, IEEE Pervasive Computing **4** (2005), no. 3, 55–60.

[MV05] Aqeel Mahesri and Vibhore Vardhan, *Power consumption breakdown on a modern laptop*, Power-Aware Computer Systems (2005), 165–180.

[NL95] William Newman and Michael Lamming, *Interactive system design*, Addison-Wesley, Boston, MA, USA, 1995.

[OS89] Alan V. Oppenheim and Ronald W. Schafer, *Discrete-time signal processing*, Prentice-Hall, 1989.

[OVLdL05] Veljo Otsason, Alex Varshavsky, Anthony LaMarca, and Eyal de Lara, *Accurate GSM indoor localization*, Proc. Intl. Conf. on Ubiquitous Computing (UbiComp), September 2005, pp. 141–158.

[PCB00] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan, *The Cricket location-support system*, Proc. Intl. Conf. on Mobile Computing and Networking (MobiCom), 2000, pp. 32–43.

[PCC+10] Jun-geun Park, Ben Charrow, Dorothy Curtis, Jonathan Battat, Einat Minkov, Jamey Hicks, Seth Teller, and Jonathan Ledlie, *Growing an organic indoor location system*, Proc. Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys), 2010, pp. 271–284.

[PM96] John G. Proakis and Dimitris G. Manolakis, *Digital signal processing*, Prentice-Hall, 1996.

[PSZ+07] Chunyi Peng, Guobin Shen, Yongguang Zhang, Yanlin Li, and Kun Tan, *BeepBeep: a high accuracy acoustic ranging system using COTS mobile devices*, Proc. Conf. on Embedded Networked Sensor Systems (SenSys), 2007, pp. 1–14.

[QBCN11] Chuan Qin, Xuan Bao, Romit Roy Choudhury, and Srihari Nelakuditi, *TagSense: A smartphone-based approach to automatic image tagging*, Proc. Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys), June 2011, pp. 1–14.

[RGMN06] Parthasarathy Ranganathan, Erik Geelhoed, Meera Manahan, and Ken Nicholas, *Energy-aware user interfaces and energy-adaptive displays*, IEEE Computer **39** (2006), no. 3, 31–38.

[RHM+02] Judy A. Roberson, Gregory K. Homan, Akshay Mahajan, Bruce Nordman, Carrie A. Brown, Richard E. Webber, Marla C. McWhinney, and Jonathan G. Koomey, *Energy use and power levels in new monitors and personal computers*, Tech. report, Lawrence Berkeley National Lab, Berkeley CA., July 2002.

[RN03] Stuart Russell and Peter Norvig, *Artificial intelligence: A modern approach*, 2nd ed., Prentice Hall, 2003.

[RSW05] Daniel M. Russell, Norbert A. Streitz, and Terry Winograd, *Building disappearing computers*, Commun. ACM **48** (2005), 42–48.

[RWM+04] Judy A. Roberson, Carrie A. Webber, Marla C. McWhinney, Richard E. Brown, Margaret J. Pinckard, and John F. Busch, *After-hours power status of office equipment and inventory of miscellaneous plug-load equipment*, Tech. report, Lawrence Berkeley National Lab, Berkeley CA., January 2004.

[SD05] James Scott and Boris Dragovic, *Audio location: accurate low-cost location sensing*, Proc. Intl. Conf. on Pervasive Computing, 2005.

[Sim79] James A. Simmons, *Perception of echo phase information in bat sonar*, Science **204** (1979), 1336–1338.

[SOM+08] Alex Shye, Berkin Ozisikyilmaz, Arindam Mallik, Gokhan Memik, Peter A. Dinda, and Robert P. Dick, *Learning and leveraging the relationship between architectural-level measurements and individual user satisfaction*, Proc. Intl. Symposium on Computer Architecture (ISCA), June 2008.

[SPS+08] Alex Shye, Yan Pan, Ben Scholbrock, J. Scott Miller, Gokhan Memik, Peter A. Dinda, and Robert P. Dick, *Power to the people: Leveraging human physiological traits to control microprocessor frequency*, Proc. Intl. Symposium on Microarchitecture (MICRO), November 2008.

[TDDM09] Stephen P. Tarzia, Robert P. Dick, Peter A. Dinda, and Gokhan Memik, *Sonar-based measurement of user presence and attention*, Proc. Intl. Conf. on Ubiquitous Computing (UbiComp), September 2009, pp. 89–92.

[TDDM10] Stephen P. Tarzia, Peter A. Dinda, Robert P. Dick, and Gokhan Memik, *Display power management policies in practice*, Proc. Intl. Conf. on Autonomic Systems (ICAC), June 2010.

[TDDM11] ———, *Indoor localization without infrastructure using the acoustic background spectrum*, Proc. Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys), June 2011, pp. 155–168.

[Thr02] Sebastian Thrun, *Robotic mapping: a survey*, Exploring Artificial Intelligence in the New Millenium, Morgan Kaufman, 2002, also CMU-CS-02-111.

[TMV04] Jeanette A. Thomas, Cynthia F. Moss, and Marianne Vater (eds.), *Echolocation in bats and dolphins*, University of Chicago Press, 2004.

[War98] Andrew Ward, *Sensor-driven computing*, Ph.D. thesis, Corpus Christi College, University of Cambridge, Cambridge, UK, August 1998.

[WB97] Mark Weiser and John Seely Brown, *The coming age of calm technolgy*, Beyond calculation, Copernicus, New York, NY, 1997, pp. 75–85.

[WH08] Oliver Woodman and Robert Harle, *Pedestrian localisation for indoor environments*, Proc. Intl. Conf. on Ubiquitous Computing (UbiComp), 2008, pp. 114–123.

[WHFG92] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons, *The Active Badge location system*, ACM Trans. Information Systems **10** (1992), no. 1, 91–102.

[YA05] Moustafa Youssef and Ashok Agrawala, *The Horus WLAN location determination system*, Proc. Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys), June 2005, pp. 205–218.