ENABLING MOBILE MICROINTERACTIONS

A Thesis
Presented to
The Academic Faculty

by

Daniel L. Ashbrook

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
May 2010

ENABLING MOBILE MICROINTERACTIONS

For Sally.

ACKNOWLEDGEMENTS

I owe many thanks to many people for their help, patience and kindness to me during the lengthy process of starting and finishing my PhD.

First and foremost, thanks are due to my wife, Sally—I couldn't have done it without her. She kept me sane, happy and fed during the whole process, and without her kindness, patience, love and encouragement I would probably still be debating my thesis topic.

My parents provided much love and moral support throughout the long process of my three degrees. Neither they nor I foresaw a 15-year, 3-graduation affair when I entered Georgia Tech in 1995, but they've been there for me through thick and thin.

I would be somewhere—and someone—else today without the support of Thad—my advisor, mentor, and friend. I aspire to one day be able to work as hard as he does while still remaining as genial and unflappable as he is.

The members of the Contextual Computing Group have always been there with ready advice, especially when it comes to presentations. The mobile HCI contingent—Kent Lyons, James Clawson, and Nirmal Patel—has been invaluable with ideas, discussion, and help.

Finally, Foo and Olivia have provided much-needed stress relief, and have helped me to always remember what's most important in life: dinner time and head-scratching, in that order.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Summary

While much attention has been paid to the usability of desktop computers, mobile computers are quickly becoming the dominant platform. Because mobile computers may be used in nearly any situation—including while the user is actually in motion, or performing other tasks—interfaces designed for stationary use may be inappropriate, and alternative interfaces should be considered.

In this dissertation I consider the idea of *microinteractions*—interactions with a device that take less than four seconds to initiate and complete. Microinteractions are desirable because they may minimize interruption; that is, they allow for a tiny burst of interaction with a device so that the user can quickly return to the task at hand.

My research concentrates on methods for applying microinteractions through wrist-based interaction. I consider two modalities for this interaction: touchscreens and motion-based gestures. In the case of touchscreens, I consider the interface implications of making touchscreen watches usable with the finger, instead of the usual stylus, and investigate users' performance with a round touchscreen. For gesture-based interaction, I present a tool, MAGIC, for designing gesture-based interactive system, and detail the evaluation of the tool.

CHAPTER I

INTRODUCTION

In this dissertation I consider the idea of *microinteractions*—interactions with a device that take less than four seconds to initiate and complete. Microinteractions are desirable because they may minimize interruption; that is, they allow for a tiny burst of interaction with a device so that the user can quickly return to the task at hand.

Much of the foundational work leading up to my conception of microinteractions was undertaken by Antti Oulasvirta and colleagues. Oulasvirta *et al.* investigated the fragmentation of attention in mobile situations, discovering that, to maintain attention on both the mobile device and the world around them, attention to the mobile device broke into bursts of four to eight seconds [69]. Oulasvirta recommends that designers should "put effort to shorten interaction units (down to less than five seconds)" [70]. I believe that—if possible—the *entire interaction* should be completed in under four seconds, enabling the user to return to the task at hand without any further interruption.

There are two stages to device usage that contribute to the "microness" of an interaction: access time and usage time. Access time encompasses the time it takes to get the device ready to use for its intended purpose; for a mobile phone, it includes retrieving the phone from its storage location (such as pocket, bag or holster) and navigating to the desired application. Usage time is the amount of time the device is actually being engaged for the purpose at hand, and is application dependent: checking tomorrow's weather might only take a second, while composing an email could take several minutes. Because of this dependence, I only consider access time.

One useful metric for considering access time for a device is the ratio of access time to usage time. The larger the ratio, the more need for a decrease in access time. Consider the scenario of checking tomorrow's weather: if it takes five seconds to retrieve one's mobile phone and start up the weather application, but only a half second to note the icon and high temperature, the ratio is $5 : 1/2 \rightarrow 10$, implying that the access time in this case should be sped up considerably. On the other hand, if the task is to compose a text message, the ratio might be $5 : 30 \rightarrow 1/6$, which is appropriately small;

One potential objection to the idea of microinteractions is whether there are truly enough tasks that could be made "micro". I informally investigated this question by looking at the default set of applications that came installed on the first-generation Apple iPhone (circa 2006). Of the thirteen applications, eight could conceivably be used in four seconds or less. The following list details the eight "fast" applications and describes the quick use case for each:

iPod *changing tracks, changing volume, playing and pausing, shuffling*

Phone *quickly dialing a favorite number, changing volume, hanging up, answering call waiting*

Weather *checking current and predicted weather conditions*

Calendar *looking at one's upcoming schedule*

Stocks *retrieving information on current stock conditions*

Clock *snoozing or turning off alarms*

SMS *quickly reading a received message*

Calculator *making a quick calculation*

Unfortunately, few of these applications are currently truly "micro". The iPod and iPhone applications may be controlled by a headphone remote cord, and when a single SMS has been received it is displayed on the phone's lock screen. There is much potential, however, to enable microinteractions for many devices and many applications.

In this dissertation, I concentrate on methods for applying microinteractions through wrist-based interaction. I consider two modalities for this interaction: touchscreens and motion gestures. In the case of touchscreens, I consider the interface implications of making touchscreen watches usable with the finger, instead of the usual stylus, and investigate users' performance with a round touchscreen. For motion gesture-based interaction, I present a tool, MAGIC, for designing gesture-based interactive system, and detail the evaluation of the tool.

My thesis statement is as follows:

1. Wrist-mounted sensors can be used to create gesture-based interfaces where the access interactions occur in under four seconds.

2. Providing interaction designers with databases including peoples' everyday motions allows them to create gestures with fewer false positives than can be created without the databases.

1.1 Contributions

The work I present in this dissertation makes several contributions. Supporting mobile microinteractions by reducing access time and avoiding push-to-activate (using a button to "unlock" a device before providing input) will affect everyday interactions with mobile devices. By allowing more efficient usage, users will be more likely to use the functionality that the device offers, and speeding up access time for the many daily uses that devices are put to will have a large overall effect.

The general contributions this work makes include the following:

1. *The introduction of the concept of microinteractions.* As discussed above, a microinteraction is an interaction with a device taking under four seconds to complete. **Chapter 2** discusses some of the motivating work behind the idea of microinteractions.

2. *A study quantifying the access time properties of mobile devices stored in the pocket, in a belt holster, and mounted on the wrist, while standing and walking.* Although one would naturally expect that a watch is faster to access than a pocket or holster, I quantify how much faster it is and break down the access time into several measurable steps. This study, presented in **Chapter 3**, also indicates that people are generally

capable of simple interactions with a touchscreen while performing pedestrian navigation.

3. *The introduction of new interaction techniques for the wrist.* The wristwatch is a piece of technology that is well-understood by the population at large, and that is starting to garner interest as a platform for interaction. Most existing methods for interacting with the wristwatch are, however, button-based, often leading to unnecessary complexity. In **Chapters 4** and **5**, I propose two new methods for interacting with a wristwatch that could lead to simpler or more efficient interactions with multifunctional wristwatches.

My next contribution relates to touchscreen wristwatches with round faces:

4. *A study determining the optimal on-screen button size for three interaction types on a round touchscreen watch.* As discussed later, a round display suggests an interaction region around the edge and a separate center area. Given the edge-based interaction area, what type of touch interaction should be supported, and how many buttons may be accommodated? This study, presented in **Chapter 4**, compares three types of touches—tapping, sliding in a straight line, and sliding along the rim—and determines the optimal button size for each.

The final set of contributions is related to gesture-based interactions:

5. *A software tool for gesture design, that assists in creating low-false positive gestures.* In **Chapter 5** I introduce MAGIC, a tool to help interface designers create motion gestures. I discuss the workflow of gesture creation and how the software supports it. One of the features of MAGIC is the *Everyday Gesture Library* (EGL), which assists gesture designers in designing freeform gestures that are less likely to be falsely triggered by a person's everyday movements than gestures designed without the assistance of the EGL.

6. *Strategies used by non-domain experts in designing motion gestures* using MAGIC are discussed in **Chapter 6.** A wide variety of strategies were used both for memorability

purposes and to make the gestures more distinguishable from each other.

1.2 Dissertation Organization

In Chapter 2, I discuss work related to my thesis, including work related to microinteractions in general, as well as to touchscreen wristwatches, gesture, and interaction design tools. Chapter 3 presents a study on the amount of time required to access devices placed at various locations on the body, and argues that the wrist is a good location for interactive technology. Chapter 4 explores one method of implementing wrist-based interaction, through a round touchscreen watch. Chapters 5–7 present and discuss MAGIC, a tool for helping gesture designers create systems in which the gestures do not conflict with each other and have low false positive rates. Finally, Chapter 8 concludes the dissertation.

CHAPTER II

RELATED WORK

2.1 Microinteractions & Mobility

In a 1968 paper, Miller characterized the delays inherent in human-computer "conversation" [61]; that is, the amount of time that it takes for a computer to respond to a human's input, and how the human will react to the delay. He recommended, in most cases, that delays of no more than two seconds be allowed:

> A general rule for guidance would be: For good communication with humans, response delays of more than two seconds should follow only a condition of task closure [a task or sub-task being finished] as perceived by the human, or as structured for the human.

In Miller's era, computers were fixed in one place, and the sole purpose of a person being in front of one was to accomplish a particular task. Nearly 40 years later, Oulasvirta *et al.* considered a different environment, in which users are mobile [69]. Participants in their study were performing multiple tasks while waiting for a response from the computer (in this case a mobile phone), including navigating busy streets and metro platforms, watching for a particular metro stop, and eating and carrying on conversation. In these environments of multiple attentional demands, participants took four to eight seconds at a time to look at the device before returning attention to the environment.

In both the Miller and Oulasvirta cases, users are waiting for output, rather than performing input; it is only *after* requesting a response from the system that the waiting time of two or four to eight seconds occurs. Figure 1 illustrates the cycle of user input/computer response; in the figure, the user's waiting time is represented by the upper arrow, where the computer is performing some kind of processing, and the user must wait for a response. In Miller's examples, users were waiting for responses to a wide variety of queries to mainframe

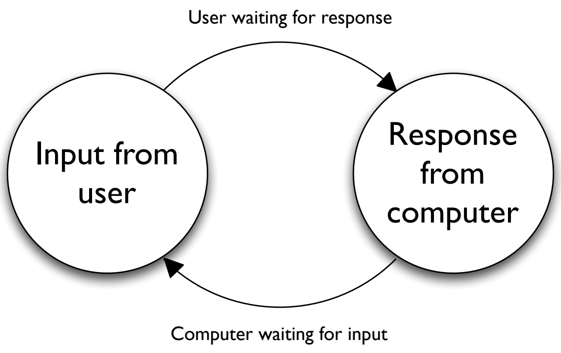


Figure 1: The human/computer response cycle, in which the computer waits for input from the user, and then the user waits for a response from the computer.

systems, including database queries and graphical model rendering. In Oulasvirta *et al.*'s study, participants were waiting for network-bound web page loading.

My work in this dissertation largely concentrates on the upper arrow, which concerns what the *human* is doing. The computer is patient and can wait all day for the user; when the user decides to use the system and enters the leftmost state, it should be possible to exit that state again very quickly. That is, the user should have the opportunity to quickly and easily issue a command, and then attend to other tasks while the computer completes its part of the cycle.

What determines whether a particular interaction with a device is a microinteraction? Microinteractions both encompass and break Figure 1's cycle. Figure 2 illustrates this: the user performs input, and once the computer has responded, the interaction is effectively finished. If the entire flow illustrated can be completed in a particular time—lacking other data, I use Oulasvirta's lower time of four seconds—then the interaction can be considered to be "micro".

The actual time that the interaction takes is dependent upon all parts of the cycle: the user's ability to quickly communicate to the computer what it is she would like it to do, the computer's ability to speedily execute the desired task, and the ability of the user to consume the output (if any) from the task. The "input from user" part of the cycle can be thought of as the *gulf of execution*, and the "response from computer" part as the *gulf of evaluation* [32]. The goal of microinteractions—as in all HCI—is to reduce the gulfs; the

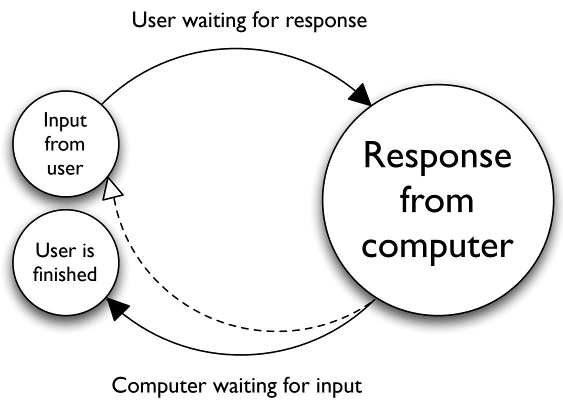


Figure 2: The microinteraction cycle, in which the computer waits for input from the user; once the user receives a response from the computer, the interaction may be finished (solid line). In a multiple-step interactive task the cycle may repeat one or more times (dashed line).

goal of my dissertation work in particular is to reduce the gulf of execution.

To address this challenge, especially with mobile devices, I consider *access time*—the amount of time it takes to get a device ready to use. I also consider the *first input* to a system, which in the case of microinteractions is ideally the *only* input. One way to reduce access time is by making the device extremely accessible. Section 2.1.1 discusses some relevant work on the subject of getting to and using mobile devices, including some justification for using the wrist as an ideal platform for microinteractions.

If the wrist makes an excellent platform for interaction, what kind of interaction should take place upon it? In this dissertation, I describe two approaches: touch and gesture. Because part of a microinteraction can involve several user-computer response cycles (for example, specifying which day of the calender to show by scrolling through a list), an efficient method of interacting with the device is necessary. I investigated the properties of a touchscreen watch with a round face; work related to this idea is presented in Section 2.2.

In some cases, the dotted line in Figure 2 is never followed; this the case of an instant command. Pressing the volume button on an audio player is an instant command: the user's input—pressing the button—is immediately followed by the computer's response—raising the volume—and, unless the result is unsatisfactory, requires no further interaction. Buttons, however, can require more access time, especially if there are a lot of them; therefore, I consider gesture as a way of performing instant commands. A discussion of work related to gesture appears in Section 2.3.

Gesture creation is a challenging task, particularly for non-experts. I created a piece of software, called MAGIC, that helps interaction design professionals design and test gestures. MAGIC draws inspiration from a number of prior projects, which are discussed in Section 2.4.

2.1.1 Access Time

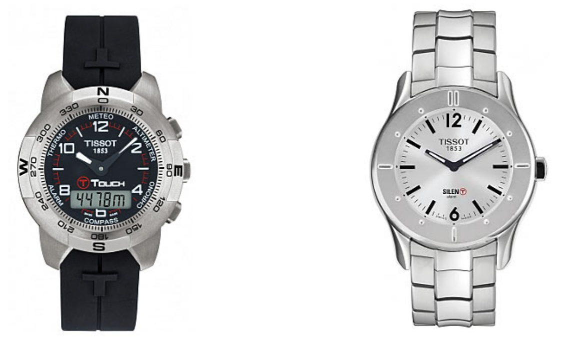
Patel *et al.* examined people's perceptions about how often they have their mobile phone nearby. Their data show that people routinely overestimate the physical availability of their mobile phones [72]. Even when the mobile phone is with its user, it may not be quickly accessible; Cui *et al.* found that 40% of women and 30% of men miss phone calls simply due to the manner in which they carry the mobile phone on their person [18]. Similarly, Starner *et al.* found correlations between an individual's decision to use or not use a mobile scheduling device (such as a day planner or PDA) and the amount of time and effort required to access and make ready the device [94]. Together, these studies suggest that the time required to access a device may be an important property affecting mobile use.

Kristoffersen and Ljungberg noted a variety of problems with portable devices that combined to prevent activity from simply "taking place"—instead, users of the technology had to "make place" to use the device [44]. The biggest problems they found appeared to be related to the form-factor of the devices used, which were PDAs: users needed two hands to use the device, but often only one or no hands were free; and frequently they needed to put down the device to use it—either to have a surface to type or write on, or to refer back to the display—but no surface was available for this purpose. These results suggest that, for some purposes, a wrist-based platform may be more practical.

Chapter 3 discusses a study I worked on that measured the access time required for a wrist-mounted device, as compared to a device in a pocket or in a belt holster.

2.2 Touchscreen Wristwatches

Despite the increasing functionality in wrist-worn technology, there has not been much academic work on wristwatch interfaces, and the literature is particularly sparse in the area of round touchscreen watches. Two commercial examples of round-faced watches with touch capability—though not with round touchscreens—are the Tissot T-Touch and Silen-T wristwatches[1], shown in Figure 3. Both watches feature a "push-to-touch" mechanism, designed to avoid accidentally activating functionality. The T-Touch is activated by holding a finger to the touch-sensitive crown for longer than one second, and the Silen-T is activated by pressing in the crown (which is a button). This push-to-activate mechanism is a recurring feature of many sensor-based mobile interaction systems [40, 56, 91].



(a) (b)

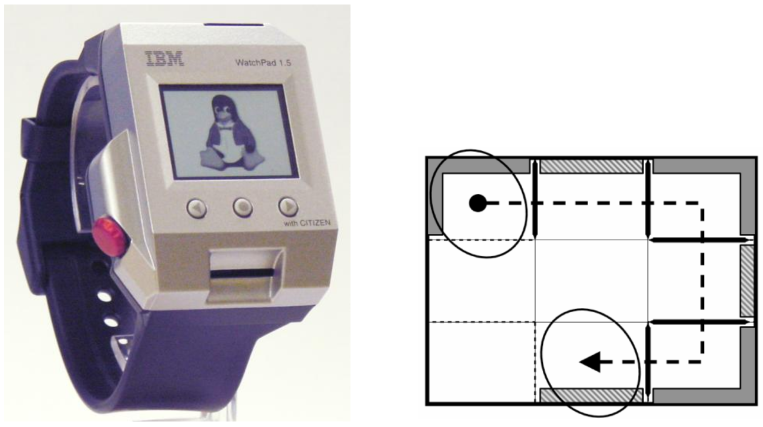
Figure 3: The Tissot T-Touch (a) and Silen-T (b) touch-sensitive watches. The T-Touch's touch mechanism is activated by touching the crown (next to the **E** on the bezel) for longer than one second; the areas on the crystal labeled THERMO, METEO, ALTIMETER, CHRONO, COMPASS, and ALARM then become sensitive, activating various functionality when touched. The Silen-T also features a touch-sensitive crystal, which is activated by pressing the crown. The watch then vibrates when the user's finger is run clockwise around the face, with a long vibration when the finger is at the current hour position and short vibrations at the minute position.

Despite the paucity of literature on round touchscreen watches, there has been work on rectangular touchscreen watches. IBM's Linux-based WatchPad prototype (Figure 4(a)) incorporates a four-zone touchscreen, and Raghunath *et al.* implemented some simple interfaces for it [79]. However, because of the low resolution of the touchscreen, most of the

[1]`http://tissot.ch`

applications depended upon the scrollwheel included on the watch (visible in Figure 4(a) on the left side of the watch).

Blaskó further extended the IBM work, creating a rectangular prototype—although non-watch-based—with nine touch-sensitive zones (Figure 4(b)) that allowed for virtual scroll wheels [26], sliding, and gestures [25]. In his interfaces, Blaskó used the bezel at the edges of the display as a "tactile landmark" to inform the user in an eyes-free manner of their current location on the touchscreen. Unfortunately, the interfaces were only evaluated in stationary situations, and no consideration was given to the potential issue of accidental activation.



(a) (b)

Figure 4: The IBM WatchPad (a) and Blaskó's interface for a variation with nine touch-sensitive areas.

One commercial (although not commercially successful) example of a touchscreen watch was the Fossil/Abacus WristPDA, which was based on PalmOS and had a 160x160 pixel touchscreen. While its main menu showed only four icons (Figure 5(a)) and could conceivably be operated with a fingertip, other application screens were tiny enough (Figure 5(b)) to require the use of a stylus built into the watch band (Figure 5(c)).

Baudisch and Chu investigated touch interaction with wristwatch-sized and smaller screens, but through the unique approach of touch input on the *back* of the device [6]. Their input device was a small (2.4" diagonal) rectangular screen with a touchpad on the

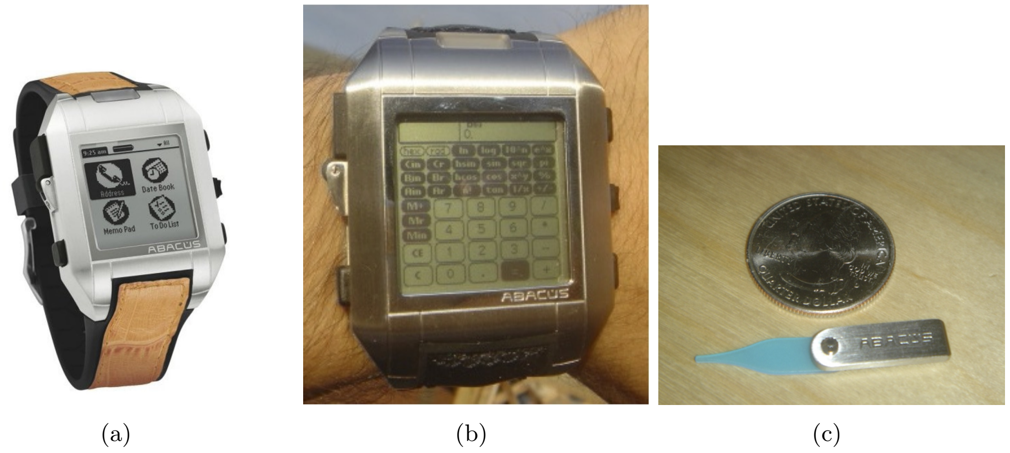


Figure 5: The Fossil/Abacus WristPDA. (a) shows the application menu, with icons large enough to push with a fingertip; (b) shows the calculator application with tiny buttons; (c) illustrates the scale of the included stylus that is normally tucked into the watch band. It is likely that the stylus or another pointed instrument would be needed to effectively use the calculator application.

back. Although they proposed that such an input technique might be usable for wristwatches, they did not prototype or test the wrist interaction.

While not wristwatch-based, there have been a number of publications on the general topic of circular interactions. Several techniques investigate menu selection; the most well-known are the Pie Menu [14], which uses a round menu with wedge-shaped buttons (Figure 6(a)), and its extension, Marking Menus [46]. These ideas have been extended in numerous ways: FlowMenu [28], for example, allows in-place hierarchical menu selection, command parameter entry and text input; earPod [109] used a pie-menu like design for an auditory menu (Figure 6(b)); and Gellersen *et al.* [27] developed a pie menu-like interface for wearable computers (Figure 6(c)). Scrolling is another common interaction using circular interfaces. The Radial Scroll Tool [89] allows a user to scroll through a document by making a "dial turning" motion. The Virtual Scroll Ring [63] is a similar idea. As well as moving through documents, scrolling widgets can be used for parameter selection: both FlowMenus and the Curve Dial [90]—the successor to the Radial Scroll Tool—allow for eyes-free parameter entry.

Text entry has also been implemented using circular interfaces. FlowMenu uses an adaptation of the Quikwrite technique [75]. Cirrin [57] uses a circular key layout and a

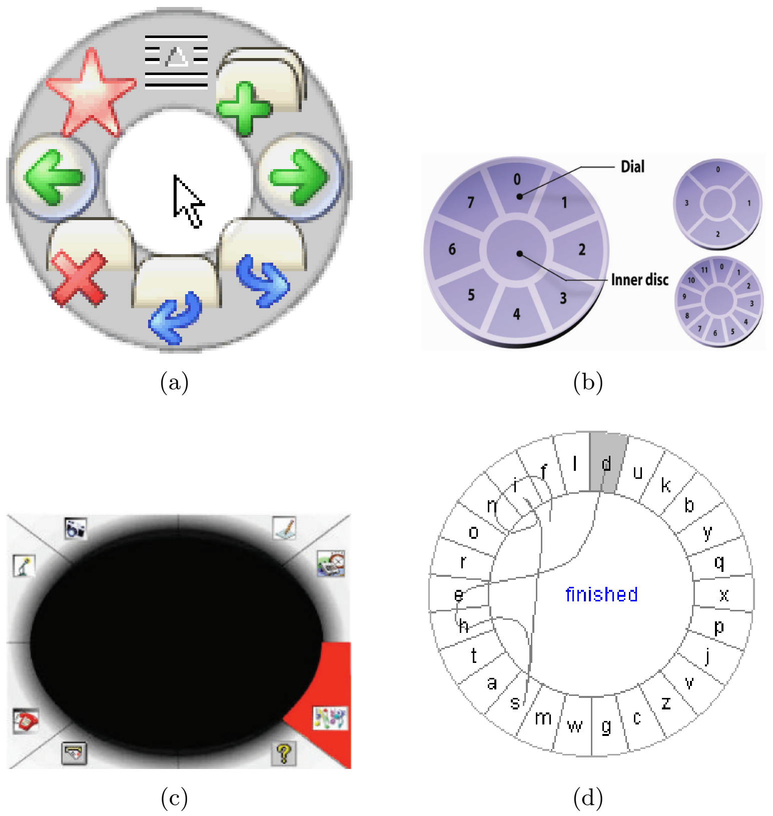


Figure 6: Interfaces using a center/rim style of interaction. (a) is an example of a pie menu [14], (b) is a conceptual visualization of the earPod menu system [109], (c) is a pie menu-like selection system designed for see-through head-mounted displays (the central black area will appear to be transparent to the wearer) [27], and (d) illustrates Cirrin [57].

continuous gliding-style interaction to enter text (Figure 6(d)).

Much of the literature on circular-style interfaces defines, whether explicitly or implicitly, two regions of interaction: the rim and the center. While these areas also make sense for a round touchscreen watch, I have encountered a lack of data on appropriate widget sizes for wristwatch-sized screens.

2.3 Gesture

In this section, I discuss some of the relevant previous work in motion gesture. As opposed to pen gesture, motion gesture involves a free-space movement with some part of the body. In the literature, the body parts most frequently used for gesture are the hand and fingers [1, 9, 19, 31, 35, 36, 45, 48, 49, 67, 73, 78, 85, 88, 96, 103, 110]. Interesting exceptions include using the head [37], arms [2, 17, 80, 86], and even feet [47] to gesture.

Hand gesture has at various points made its way to the consumer market, mostly for video gaming. The Power Glove was a 1987 peripheral for the Nintendo Entertainment System that featured crude ultrasonic tracking of the hand in 3D and finger position sensing. A similar device, the P5 Glove was released in 2002. The Nintendo Wii game system, released in 2006, uses a motion-sensitive wand, and Microsoft's upcoming project Natal features a camera-based set-top box to sense player's hand movements.

Gesture has not appeared in many consumer devices outside of video games. Multiple Apple iPod models (including the iPhone and Nano) support "shake to shuffle", and Toshiba's Qosmio G55 laptop features camera-based gesture control for multimedia. Many other companies have announced forthcoming gesture-based devices, but few have actually appeared in stores.

The rest of this section details the results of a survey of much of the literature on motion gesture. In a summary of a 1995 CHI workshop on gesture for user interfaces, Wexelblat points out a number of questions about and issues with gesture interfaces [104]; I used many of these questions as guidance when conducting the survey. Below are some of the relevant questions, quoted verbatim from the Wexelblat paper:

- *What are the "right" applications?*
- *What is the role of learning/what can we expect of the user?*
- *What is the role of feedback (especially in terms of establishing context)?*
- *Can we work with "spontaneous" gestures?*
- *Viewpoint manipulation versus object manipulation*
- *What technologies do we need to develop for gesture applications to be practical?*
- *What are the differences between gesture as a novel input mode to applications versus gesture as replacement for other input devices?*
- *How much can we do with a small set of gestures? (If it's hard to learn a large set of gestures, what's the right minimal set?)*

- *What aspects/features of applications cause us to want to use gestures?*
- *How can we map gestures into useful application action (intuition, level of abstraction, level of autonomy)?*

More specifically, for each paper I considered the following questions:

- What are the relevant features of the gestures presented? What kinds of movements are used? What body part gestures? Is social acceptability addressed? (Section 2.3.1)
- What effects do the gestures produce, or for what purpose are they intended? Do they manipulate objects or issue commands? Are they for continuous or discrete control? (Section 2.3.2)
- From the user's side: How are gestures represented? How is recognition triggered? In what kinds of situations are the gestures intended to be used? What kind of feedback is given? Who creates the gestures? (Section 2.3.3)

2.3.1 Properties of Gestures

The motions of gestures themselves have various properties: planarity, iconicity, multi-modality, part of body involved, orientation-sensitivity, reason gesture was generated (purposefulness), and social acceptability. Some of these are purely physical, and some involve the person/gesture space.

Gestures have been broadly divided into three types, by function [12]. This division generally applies to gestures made by the hands (and empty-handed), although theoretical extensions have been created for gestures made with objects as well [13]. The types are:

epistemic gestures are used for perception; they are gestures used to explore the environment by touching and manipulation. Examples include understanding the state of an object (*touching the switch to see if the garage light was left on*) or touching as a precursor to manipulation (*finding preset button #2 on the car radio*);

semiotic gestures are used for communication. They include *gesticulation* while talking and pointing to indicate objects;

ergotic gestures are intended to have an effect on the world. Picking something up, rotating an object, and pushing something are all ergotic gestures. Many gestures used for interacting with computers are ergotic as well, especially when interacting with touchscreens (pinch-to-scale and turn-to-rotate, for example).

The intent behind a person's movements is an important characteristic of gestures. While much research has been conducted on gestures that naturally occur while speaking (*gesticulation*) [60] or communicating (*semiotic gesture*) and learning (*epistemic gesture*) [12], human-computer interaction is usually concerned with purposeful gesture—either ergotic or semiotic.

The *planarity* of a gesture involves the motion through physical space of the part of the body that is gesturing. Is the motion confined to a 2D plane (or 1D line) embedded in 3D space? For example, putting your hand flat in front of you, then moving it from right to left, is a 1D gesture. Drawing a circle in the air with your finger is an example of a 2D planar gesture. An example of a 3D gesture is drawing a circle, then drawing a half-circle perpendicular to the first circle. Most gestures in the literature fall into the 2D planar category; it may be simply that users have difficulty envisioning gestures in 3D. Kela *et al.* report on a study in which users were asked to imagine and sketch gestures for controlling home appliances [38]; they found very few 3D gestures, even though participants were not constrained by the necessity of gestures actually being recognized by a computer.

Iconicity is how representative the shape or motion of the gesture is of the action that results from performing the gesture. In Western culture, *up* and *right* are associated with increasing values, while *left* and *down* are associated with decreasing values. In the Gesture Pendant project [3], moving a sideways-pointing finger up increased the volume of the stereo, while moving it down decreased the volume; similarly, moving a vertically-pointing finger right increased the television channel, while moving it left decreased the channel.

Highly iconic gestures can also involve direct manipulation. Many motion-gesture systems in the literature have focused on interaction with virtual objects on-screen or in a virtual environment [9, 19, 31, 36, 81, 88, 103, 107, 110], and use gestures directly representative of the desired effects: a rotation of the hand will rotate the virtual object, or two hands pulled apart as though stretching a rubber band will scale an object in size.

Some gestures, on the other hand, are non-iconic. Many concepts and commands simply have no good equivalent: “save document” or “check email,” for example, have recognizable visual icons, but there is no natural gestural equivalent. Another cause for low iconicity can be the expressive ability of the input mechanism: putting one’s hand in a posture with the thumb and ring finger closed and the index, middle and pinky fingers open in order to rotate an object [36] is much less iconic than looking at the object and rotating one’s hands to mime rotating the object [9].

Multimodal gestures involve more than one input to the system. Frequently, one modality is gestural and the other is not; most commonly, these systems integrate speech with hand gestures [8, 9, 31, 45, 103]; also extant in the literature are systems integrating eye tracking and hand gestures [9, 78, 103], as well as head gestures and joystick motion [37].

Somewhat wrapped up with multimodality is the *part of the body* with which gestures are made. By far the most common are hand and figure gestures[2], although systems using head gestures [37], and arm gestures [2, 80] can be found. Also fairly common are systems that involve some physical object for manipulation; frequently this is a custom sensor box [11, 34, 38, 58, 74] although the Nintendo Wii Remote is beginning to be used as well [29].

Whether or not a gesture is *orientation sensitive* depends on both the task and the algorithm. Highly iconic gestures, due to the attached meaning, will tend to have specific orientations mean specific things; an example is “up” for increasing and “down” for decreasing. Gestures used in games are frequently orientation sensitive as well: casting a line for “fishing” wouldn’t make sense if performed sideways. For less iconic tasks, non-orientation sensitive gestures may be more appropriate. The fourth generation Apple iPod Nano digital music player, for example, implements a “shake to shuffle” mechanism: when the device is firmly shaken in any direction, a random song is played.

Implementation details may also influence whether a gesture is orientation sensitive. Systems that detect large-scale motions, such as hand gestures, may factor out the influence of gravity. Other systems may, due to the sensing mechanism, be naturally orientation

[2]Due to the commonality of hand and finger gesture system, I will not cite those papers here.

independent. Examples include systems based upon finger posture [36], slight muscle movement [17], or finger tapping [21]. Indeed, aside from adding additional hardware to sense orientation, these systems *cannot* be orientation dependent.

2.3.2 Effects of Gestures

In the literature, the effects of gesturing generally fall into one of two categories: manipulating objects or controlling functions.

Object manipulation systems focus on moving, rotating, scaling and otherwise affecting objects on the screen, in a virtual environment, and in some cases, in the world. "Put-that-there", one of the earliest gesture recognition systems [8], used gesture to place objects (chosen through speech recognition) in particular places on a screen. Dannenberg and Amon created a pointing-based interface design system to allow users to manipulate virtual knobs, buttons and switches with their fingers [19]. Segen and Kumar used a 3D camera system to track a user's hand and recognize pointing, reaching and "clicking" finger postures for a 3D drawing program and a flight simulator [88]. Guo and Sharlin used Nintendo Wii controllers to manipulate a Sony Aibo robotic dog, causing it to mirror the posture of the user [29].

Perhaps due to the waning of popularity of virtual reality (VR) research, most contemporary motion gesture systems concentrate on control rather than object manipulation (although there is a large body of object manipulation research based upon tabletop gestural interaction). An early control system by Pausch and Williams used hand gesture as a sort of joystick, to control a 2D cursor which in turn generated speech-like sounds for disabled users [73]. Another system for accessibility, by Keates and Robinson, tested head gestures [37] for potential use for computer input.

The Gesture Pendant, one of the first projects I worked on [3, 92], used broad hand gestures to control household devices such as a stereo, television or thermostat. Several other pieces of research have also investigated controlling media appliances with gesture [1, 17, 38, 49, 58, 76, 83, 95]. Krum *et al.* used a Gesture Pendant as the control for a virtual earth flythrough [45]. Rachovides *et al.* used gestures to control a three-dimensional

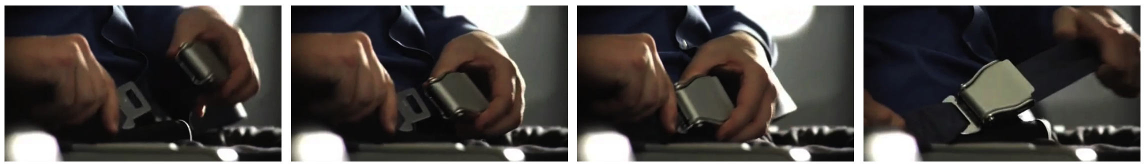


Figure 7: These stills from a Delta Airlines in-flight safety video illustrate gesture communication by *demonstration*, with an actor showing how to fasten the two parts of the seat belt.

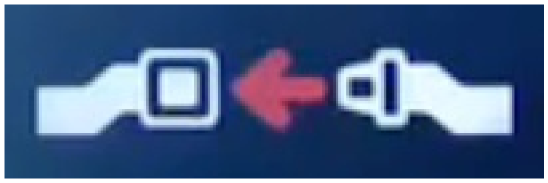


Figure 8: This airline "fasten seat belt" sign (from a Delta Airlines in-flight safety video) is an illustration of an *annotated control diagram*, showing stylized representations of the two parts to the seat belt, with an arrow indicating how they should be joined.

presentation creation system [78]. For a more abstract task, Brown *et al.* [11] implemented a simple gesture system to send vibrotactile messages between mobile phones.

2.3.3 Meta-system Considerations: User

From the end-user's perspective, there are a number of considerations about the gesture systems themselves, rather than the individual gestures. How are gestures represented such that users can learn them? How is recognition activated, and how does the system avoid interpreting every motion made by the user as a gesture? Who creates the gestures—are they dictated by the system designer, or does the user have a say?

2.3.3.1 Gesture Representation

One question is how the gestures are represented to the end user. Only some of the systems in the literature discuss how this communication takes place (however, in many of the papers I surveyed, no user tests were reported).

There are several methods of communicating gestures (adapted from a list on the Interactive Gestures Pattern Library website[3]):

Description uses language to communicate gesture. For example, Delta Airlines' in-flight

[3]`http://www.interactivegestures.com`

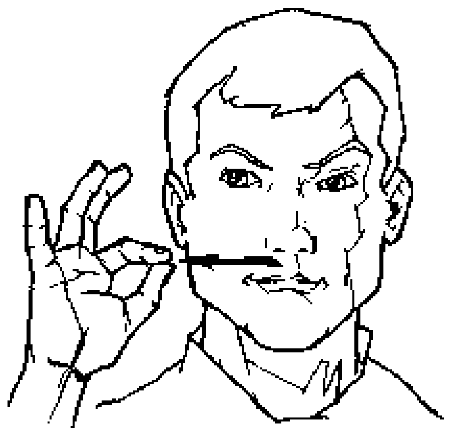


Figure 9: This image of the American Sign Language representation of the word "cat" is an example of an *annotated control diagram*.

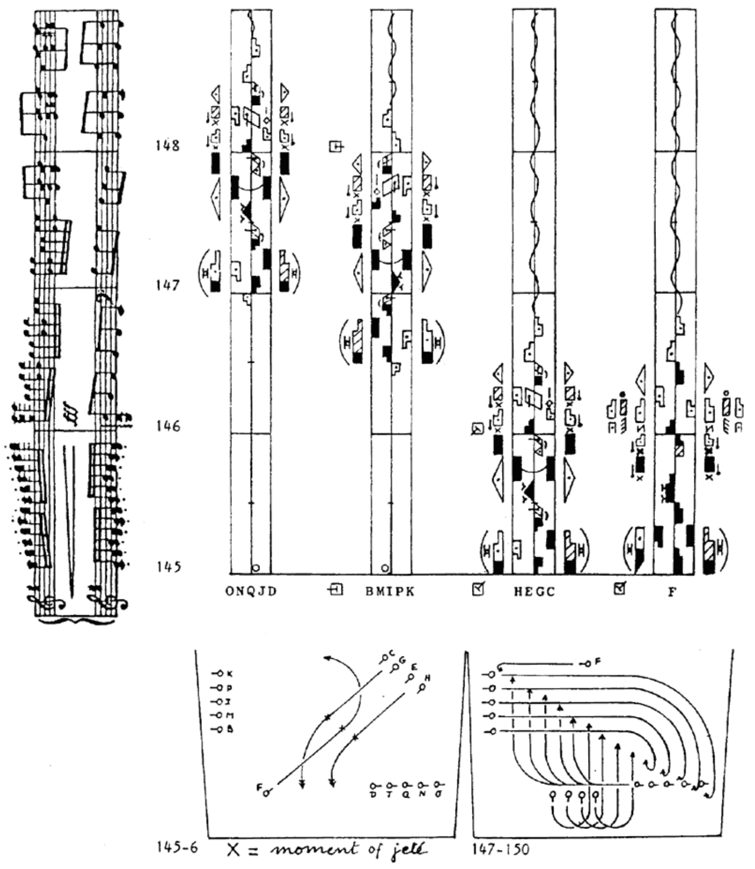


Figure 10: Labanotation, used in Laban Movement Analysis, is a complex system for representing dance movements. This image is from Rudolf Laban's 1928 work on the subject, *Schrifttanz*, or *Writing Dance*.

safety video verbally describes how to fasten the seat belt: "To fasten, insert the metal tip into the buckle and adjust the strap so it's low and tight across your lap. To release the belt, just lift the top of the buckle or press the latch to release."

Metaphor can be used to help users remember gestures. The Nintendo Wii game "Wario Ware Smooth Moves" uses metaphoric description to prompt players to hold the controller in particular ways. For example, for "the handlebar" grasp, the game instructs "Turn the [controller] sideways and grasp the ends firmly in both hands. Like riding a bicycle, perfecting this stance requires grace, steadiness, and tight shorts."

Demonstration communicates gesture directly through a a live person, video or animation; an example is a flight attendant demonstrating how to fasten an airline seat belt (Figure 7).

Diagrams use drawings—frequently composed of or including arrows—to communicate gestures; *annotated control diagrams* include an image of the object or limb with words or images describing the gesture to be made; an example is the "fasten seat belt" sign on airplanes (Figures 8 and 9).

Existing languages can quickly communicate gestures to those who know the language. For example, American Sign Language signers know the gesture for "cat" without further description; for the rest of us, an annotated control diagram is helpful (Figure 9). Similarly, Laban Movement Analysis uses complex diagrams to represent dance movements (Figure 10).

Most of the papers I surveyed used the demonstration method to communicate gestures to users: the researcher simply told and/or showed the user what to do [34, 35, 45, 76]. Several other papers used line-and-arrow diagrams (although not annotated control diagrams) to instruct users [1, 37, 68]. The rest of the papers I surveyed did not give details on how gestures were communicated.

2.3.3.2 Gesture Activation & Situation

One concern from the perspective of the end user is the "Midas touch" effect. In eye-tracking systems, the Midas touch effect [33] refers to the difficulty in determining when a user intends her gaze to activate a UI element: when the user looks at a button, was the intent to push it, or just to read its label? Speech recognition-based systems can suffer from the same difficulty; many users of speech-based automated voice response telephone systems have been frustrated when the system interprets background noise as menu selections. Some research systems solve this problem with a "push-to-talk" mechanism [56], where the system does not listen for input except for when the user is holding down a button. An alternate solution is to listen for a specific word [39]. Viewers of the television series Star Trek will be familiar with how characters on the show prefixed commands to the ship with the word "computer", as in "**Computer**, tell me Captain Picard's location!"

Gesture-based systems can also suffer from the Midas touch problem. With the Gesture Pendant, a wearable computer vision-based gesture system [91], we used a push-to-gesture mechanism to keep the system from recognizing normal conversational gesticulation as gestures. For a non-prototype system, however, pushing-to-gesture may be undesirable, as one could simply activate the desired function using the button itself, rather than making a gesture. A "gesture-to-gesture" system is a natural solution; for example, the Gesture Watch [40] used a bent wrist to signal the watch to look for hand movement. In the evaluation of my MAGIC system (Chapter 6), several participants independently came up with a gesture-to-gesture function.

One consideration that can affect the kind of activation to be used for a gesture system is the situation in which it will be used. If gestures are only to be used in constrained situations, such as changing the posture of an avatar in a VR environment [48] or controlling 3D presentation software [78], accidental activation is less of a concern. In unconstrained environments, however, such as with wearable devices [17, 40, 91] or public infrastructure [85], measures such as push-to-gesture must be taken to ensure that movements not intended for the system are not interpreted as though they are.

2.3.3.3 Gesture Definition

In the literature, the method by which gestures are defined—as well as who defines them—tends to fall between two extremes. On the one extreme are completely recognizer-based approaches, wherein the gestures are dictated by and built around the gesture recognition algorithm. Most of the papers in this category focus primarily on the introduction of a new recognition approach. On the other extreme are relatively few user-driven studies, in which recognition is largely (or completely) ignored in favor of determining what kinds of gestures are natural for and desirable to users.

The majority of papers in my survey fell into the extreme of recognizer-based approaches. In these papers, the researcher simply states—often without justification to the reader—what gestures were chosen for the experiment. Frequently, the gestures are based upon the sensor that was chosen for the experiment. For example, Costanza *et al.* created a system for subtle gesture in which upper-arm twitches sensed by EMG were used to control mobile devices [17]; the gestures possible were limited and defined by the sensor used. Likewise, with the Gesture Pendant [91], we chose gestures based on the limitations of the vision system, picking slow-moving, close-to-camera gestures. Gestures are often chosen to demonstrate a particular recognition algorithm or technology; in a 1987 CHI paper, Zimmerman *et al.* introduce an early glove-based gesture system [110], discussing applications and gestures in the context of illustrating the capabilities of the device.

Often researchers choose gestures based on their iconicity, although this choice frequently goes unevaluated by users. For example, Rachovides *et al.* created a hybrid gaze/gesture system to control a presentation creation application [78]. Many of their gestures were chosen to be iconic and, in some cases, metaphoric (see Section 2.3.3.1); for example, exiting the program involves pantomiming closing a box. Sometimes gestures are chosen by virtue of being directly related to the task: Guo and Sharlin created a Wii controller-based application in which a Sony Aibo robotic dog mimics a user's posture, based on how the user holds the two controllers [29].

The other extreme of gesture design is the highly user-focused approach. Often based upon participatory design principles [84], these papers usually do not involve machine-based recognition until the gestures have been created, if at all. Kela *et al.* asked 37 participants to devise and sketch gestures for in-home tasks such as VCR control [38]; the gestures were implemented and tested, but not by the users who suggested them. Nielsen *et al.* elicited gestures from participants, analyzing them for ergonomic properties and testing them for memorability [67]; algorithmic recognition of the produced gestures was not addressed. Wobbrock *et al.* performed a similar study to create gestures for manipulating objects in a tabletop environment [107].

2.4 MAGIC Related Work

MAGIC (introduced in Chapter 5) was influenced by gesture systems, programming-by-demonstration tools, and other interactive system building software.

2.4.1 Gesture Systems

A primary piece of prior work is *quill*, an interactive system for designing pen gestures, described in Long's dissertation [54] and papers [53, 52]. *quill* allowed users to create pen gestures by example, and offered automated advice on improving the gestures. In the design process for *quill*'s predecessor, *gdt*, Long found several issues with gesture design: some users did not realize that two similar gesture classes might conflict with each other; users didn't understand how the gesture recognizer worked; and finding and fixing problems with recognition was very difficult. To help alleviate this issue, *quill* included feedback such as goodness of gesture and used language to communicate issues; however, Long states that "many participants did not understand the suggestions". He goes on to posit that perhaps "the suggestions can be made more accessible. . . by using more diagrams".

MAGIC takes lessons from both *gdt* and *quill*. In *gdt*, colored matrices were presented to show the distances between classes and how examples are classified; Long reported that participants found these tables overwhelming and did not use them. MAGIC presents the same information, but in several ways that may be less confusing to users. Learning from *quill*'s problems with language, MAGIC presents more information graphically to help problems with conflicting gestures stand out.

SUEDE, a system for designing speech-based user interfaces [41], featured a design/test/analyze methodology that was a source of inspiration for MAGIC. Under this paradigm, SUEDE explicitly divides the interface into three parts. In the design phase, UI designers create a speech-response interface script. The test phase offers a Wizard-of-Oz interface to allow designers to quickly test the interface. The analysis interface allows designers to reflect on the tests and determine how the next phase of design will proceed.

MAGIC concentrates design within the Creation tab, but has testing modalities in both the Gesture Testing and EGL tabs. Analysis is available in all situations by using the visualizations of intra- and inter-class variability. SUEDE also gives justification for lowering the complexity of recognition-based interfaces, explicitly encouraging users to explore the design space rather than experiment with parameters.

2.4.2 Interaction Design Tools

MAGIC shares some features with sensor-based programming-by-demonstration (PBD) systems and other interaction design tools.

An early gesture-based design tool was described in a 1989 paper by Dannenberg and Amon [19]. Their Gestural Interface Designer (GID) allowed users to use their fingers to manipulate onscreen objects to design a "physical" interface on the screen including knobs and buttons responsive to pointing and gesture. MAGIC follows GID's example by incorporating gesture recognition as an integral part of the design process, rather than data to be collected elsewhere.

Crayons [22] is an interactive classifier training system that takes in pre-recorded images and allows users to interactively specify which classes various parts of an image should be classified as. Much like MAGIC, Crayons explicitly encourages the user to iterate by providing immediate feedback on system performance; however, Crayons is focused on classifier creation and does not consider end-user usage.

Eyepatch [59] allows users to experiment with many computer vision classifiers and build an interactive system based upon them, and includes basic gesture recognition functionality.

Exemplar, a more open-ended PBD system [30], allows users to prototype activities based on the recognition of time-series sensor data, including accelerometers. Exemplar's approach is complementary to MAGIC's: Exemplar supports multiple sensors and integrates with external hardware to create working systems, and would benefit from MAGIC's visualizations, support for retrospection, and the ability to use the EGL to ensure that everyday activities do not trigger unwanted functionality.

a CAPella allowed end-users to program complex context-aware applications by demonstrating the activities to be sensed to the system [20]. *a CAPella* could have benefited from an EGL-type collection of data, as training the system to enact the required behaviors could take multiple days.

EnsembleMatrix [97] uses visualization to help users understand the effect of different machine learning classifiers. It is, however, focused on domain experts, and as such concentrates on explicating the confusion matrices of multiple classification algorithms to users. The approach used would work well for expert users of MAGIC when more complex gesture recognition is incorporated.

2.5 Conclusion

In this chapter, I have discussed some previous research relevant to microinteractions, wristwatch interfaces, gesture and MAGIC. The next chapter presents a larger piece of related work: a study I performed with colleagues, investigating the access time for devices located at various parts of the body, and in different mobility conditions.

CHAPTER III

QUICKDRAW

This chapter is adapted from a paper presented at the SIGCHI conference on Human Factors in Computing Systems (CHI) in April of 2008 [4]. The paper was co-authored by myself, James Clawson, Kent Lyons, Nirmal Patel and Thad Starner.

The Quickdraw study provides quantitative data about the amount of time required to access a device in three positions on the body, and in two mobility conditions. The study is important for several reasons:

- *The quantification of access time for devices while standing and walking.* This information can be used as a guide to designers of mobile devices and software, to help them make decisions involving access time; for example, how long to vibrate a phone before activating the ringer.
- *An experimental design* that can be used for other studies of the same type, to test access time with different types of devices or devices carried in different locations.
- *Measurements of the stages of device access.* When retrieving a device and preparing it for use, we discovered that several stages occur. Being able to measure the time spent in each of these stages gives better information to the designer as to where improvements to the device may be made.
- *Quantitative determination of the wrist as an ideal interaction location.* Although not a surprising result, we determined the wrist as the fastest-to-access location of the three we studied. This result provides a solid foundation upon which further wrist-based interaction may rest.

3.1 Motivation

Mobile phones have become ubiquitous, with the number of subscribers worldwide predicted to pass 4.6 billion in late 2009[1]. It is not uncommon for mobile phones to be the first object with which people interact in the morning and one of the last things with which people interact in the evening [16]. Despite the importance of these mobile devices in everyday life, little empirical data has been published on many fundamental usage properties.

One of these underreported-upon properties is *access time*—the amount of time it takes a user to get to a device for the purpose of interacting with it. To take the mobile phone as a familiar example, the access time is how long it takes to get the phone from its storage place—such as a pocket, holster, or bag—and become ready to interact with the device by orienting it properly.

In this work, we examine and quantify two important factors that could impact access time: on-body placement and user mobility. Specifically, we focus on the effects of *walking* versus *standing* when accessing a touch screen interface mounted *on the wrist*, *on the hip*, and *in the pocket*. This work is not intended to influence users to change their behavior with respect to mobility or device storage, but to allow designers a better understanding of how mobility and placement affect access time.

3.2 Experiment

To explore how body placement and mobility influence the time needed to respond to and access a mobile device, we examined two independent variables with a 2x3 Latin-square within-subjects study design. The first variable is the mobility of the participant and the second is the placement of a device on the body. Although our interest in mobile devices is broader than telephones, we used a mobile phone throughout the study in order to keep the interaction consistent between conditions.

Our two mobility conditions are *standing* and *walking*, chosen because people on-the-go are likely to be in one of these two states much of the time. Our three on-body placement conditions were *in the pocket*, *on the hip in a holster*, and *on the wrist*, reflecting common

[1] `http://www.portioresearch.com/Handbook09-14.html`

placements for current mobile electronics.

During the *standing* condition we instructed participants to stand in a corner of our lab to minimize visual distractions that might interfere with access time.

For the *walking* condition, participants were instructed to walk at a normal pace around a track constructed in our laboratory (Figure 11). The track was approximately 26 meters long and was denoted with flags hanging from the ceiling with the tips 0.75 meters apart. Each flag was hung so the tip was approximately 1.6 meters above the floor. We chose to use this flag arrangement rather than floor-based track markers (as in [99]) because walking is in general a head-up task (that is, people usually look ahead some distance while walking rather than looking directly at the floor). The walking direction was counterbalanced between clockwise and counter-clockwise directions.

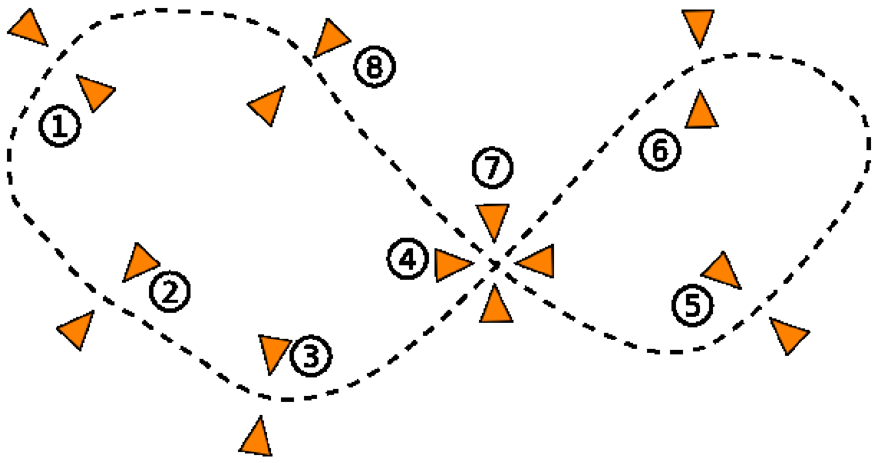


Figure 11: The path participants walked, starting at flag 1 and proceeding either clockwise or counterclockwise.

For the device placement condition, we instructed participants to put the phone into a pants or skirt pocket (after removing other items), into the manufacturer-provided holster clipped to the top of the pants or to the belt, or to attach it to the wrist with a velcro strap (Figure 12). While using an actual touchscreen watch—such as SMS Technology's M500 mobile phone watch—would have allowed for a more natural experience, we opted instead to maintain internal validity by using the same device (and therefore retaining the same display and touch input properties) for all three conditions.

To determine the amount of time required to access the phone, we asked each participant to perform a simple task. Periodically, the phone generated an alert in the form of a sound and vibration. Each participant was requested to respond to these alerts as quickly as possible. When the alert occurred, the participant retrieved the device and looked at the screen, which showed a blue box and a large number (Figure 13(a)). The participant made a mental note of the number on the display and slid the blue box to the right to unlock the phone. The participant then chose the number they had just seen from a list of four numbers (Figure 13(b)). After choosing the number, the participant returned the phone to its original position and waited for the next alert.

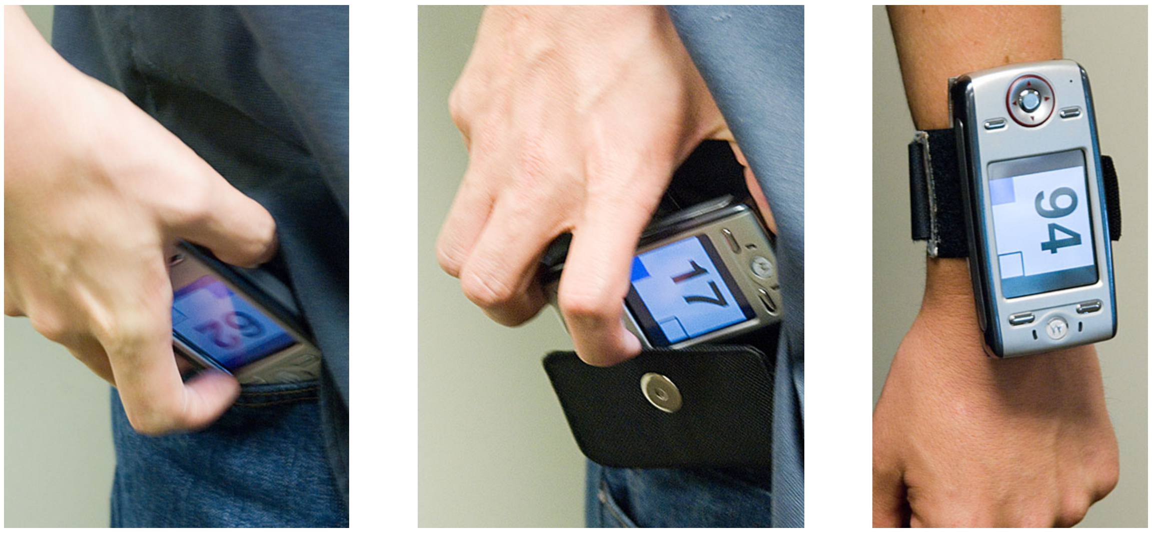

(a) (b) (c)

Figure 12: Placement: pocket (a), hip (b), and wrist (c).

3.2.1 Software and Equipment

The software for our study was implemented in Python on a Motorola E680i cameraphone running the GNU/Linux operating system. The E680i has a 320x240 color touchscreen, stereo speakers, and a number of buttons. For this study, we used only the touchscreen and deactivated all of the hardware buttons so they would not be pushed accidentally. Pygame, a Python interface to the graphics library SDL, was used to create the user interface and to log user actions.

During each condition, the operation of the software was the same. At random intervals between 20 and 40 seconds (selected from a uniform random distribution), the software generated alerts. An alert consisted of a loud mobile phone ringing sound and vibration that lasted up to 20 seconds. During the alert the software also displayed the prompt number and unlock mechanism on screen (Figure 13(a)).

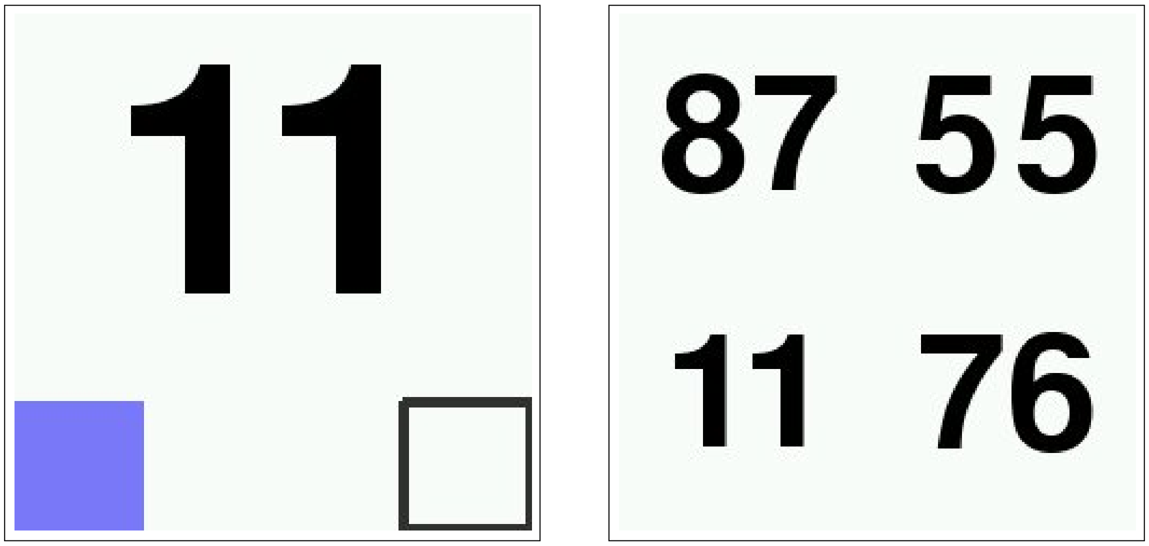
To respond to the alert, participants first unlocked the phone. Unlocking was accomplished by sliding the blue box to the right edge of the screen (similar to the Apple iPhone unlock gesture). This mechanism was implemented to prevent accidental responses while retrieving the phone from the pocket or holster. Next the phone displayed a screen consisting of a two by two grid of numbers (Figure 13(b)). The software waited for the user to select a number and logged the response. This interaction emulated common interruptions on mobile devices (for example: receiving a call, reading the caller ID, and sending the caller to voicemail). At this point the trial was complete. The phone relocked itself, and a timer was set to generate the next alert. The software logged the timestamps of each alert, the movements of the slider, and the selection of the numbers.

Because the phone had no mechanism for determining whether it was in a pocket or holster, we implemented an extremely simple light sensor using the built-in camera. Approximately four times per second, the pixel values from the camera were summed and stored. With the assumption that the holster and participants' pockets would be dark, we could detect when the phone was removed from, and replaced in, the pocket or holster. Figure 14 shows the output of the light sensor and other events logged by the software.

3.2.2 Dependent Measures

We are most interested in how device placement and mobility influences access time—the amount of time it takes for a participant to retrieve the device and respond to an alert. Figure 14 shows a timeline of a typical notification-response cycle. The points in the timeline are as follows:

1. Blank screen; participant walking track or standing.
2. Alarm starts ringing.
3. Participant pulls phone from pocket or out of holster. Light level increases to nearly



(a) (b)

Figure 13: Screen (a) is shown when an alert occurs. The participant must mentally note the displayed number and slide the blue box from the left to the target box on the right. Screen (b) is shown after the slide is completed; from the displayed four numbers, the participant must touch the number that was displayed on screen (a).

100%.

4. Participant starts to move slider.
5. Screen with four numbers is displayed.
6. Participant has picked a number; screen returns to blank.
7. Participant returns phone to pocket or holster. Light level falls to 0%.

We extracted several measurements from the timeline (numbers refer to the timeline in Figure 14):

- *Access time*: Alarm start (2) to user acknowledgment of the alarm by moving the slider (4).
- *Pocket time*: Alarm start (2) to retrieval of the device from the pocket or holster (3) (does not apply to wrist).
- *Hand time*: End of phone removal from the pocket or holster (3) to the beginning of participant's response (4) (does not apply to wrist).
- *Slide time*: Moving the slider (4 to start of 5).
- *Answer time*: Picking a number from the set of four (5).

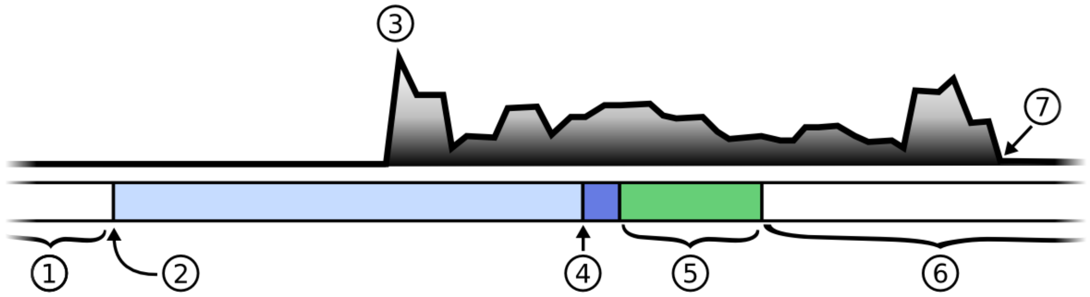


Figure 14: Timeline of events and status of brightness detector during one notification-response cycle. The top line is the percentage of light detected from the camera (complete darkness, 0% on the bottom), and the bottom is the timeline of events recorded by the phone's logging software. See the text for further details.

- *Replacement time*: Replacing the phone to the pocket or holster (start of 6 to 7—does not apply to wrist).

We define access time in this context as the time required for the participant to react to the alarm, acquire the device (either looking at the wrist or pulling it from a pocket or the holster), note the number on the screen, and touch the slider to begin sliding it.

3.2.3 Procedure

The evaluation for each participant began with the researcher presenting an overview of the study. Participants completed a consent form and a short survey about their use of mobile technology. The researcher then explained the experimental software and tasks. Each participant practiced responding to alerts three times on the phone for each of the placement conditions (pocket, hip, wrist) resulting in a total of nine practice responses. Next, the researcher familiarized the participant with the track by walking it twice in each direction; on the first lap the participant followed the researcher, while on the second, the researcher followed the participant. Finally, the researcher answered any questions from the participant, and started data collection.

Each condition consisted of a set of trials. The number of trials per condition was either five or seven, averaging to six per condition per participant. This design was selected to prevent participants from anticipating the end of a set of trials. For the walking conditions, participants were told they could slow down or stop if needed to respond to the alert quickly, but to keep walking if possible.

After the completion of the required number of trials, the phone displayed "STOP" in a large font on the screen. Participants were requested to stop where they were (if in the mobile condition) to allow the researcher to measure how far they had walked around the track.

3.2.4 Participants

We recruited fifteen participants (5 females) for our study from our academic institution. The average participant age was 24.87 years ($SD = 2.99$). Fourteen of our participants were right handed. Each of our participants owned a mobile phone and all but three had that phone with them on arrival. Six of our participants wore a wristwatch when they arrived to participate in the study. We also asked about several other devices and where the participants carried them. Table 1 provides a summary.

Table 1: Devices carried by participants and their location. "Body" refers to on-body placement of a device, such as clipping to clothing or using a wrist strap. "Bag" includes purses and backpacks.

	Pocket	Bag	Hip	Body	*Total*
Mobile phone	7	4	1	—	12
Phone headset	4	—	—	—	4
Audio player	4	5	1	1	11
Camera	6	5	—	1	12
Total	21	14	2	2	39

3.3 Results

One participant was discarded as an outlier—having taken up to 8.5 standard deviations longer than average to respond to alerts—leaving 14 participants. Each participant performed all six of the conditions. Each condition averaged six alerts per condition, for a total of 504 alert-response cycles. Table 2 shows a summary of data for each condition. For this section, we consider $p < .05$ to be significant and will only report p values for non-significant results.

A multi-way ANOVA reveals that the placement of the device has a significant effect on *access time*, but mobility is not significant ($p = .14$). There is also no significant interaction

Table 2: Statistics for mean (SD) hand, pocket and access times in seconds.

Time	Placement	Walking	Standing	*Average*
Hand	Hip	1.047 (.656)	1.339 (.728)	1.199 (.707)
	Pocket	1.109 (.647)	1.093 (.396)	1.092 (.536)
Pocket	Hip	4.355 (.906)	4.312 (.770)	4.333 (.836)
	Pocket	3.427 (.770)	3.829 (.788)	3.625 (.868)
Access Time	Hip	5.377 (.947)	5.660 (1.155)	5.518 (1.047)
	Pocket	4.414 (.904)	4.817 (.875)	4.616 (.897)
	Wrist	2.728 (.291)	2.846 (.420)	2.787 (.360)

between mobility and placement ($p = .81$). *Post-hoc* analysis of the access times using a paired Student's T-test reveals a significant difference between all three combinations of variables: hip/pocket, hip/wrist and pocket/wrist. Therefore, we have a total ordering of access time for the placement condition: wrist ($M = 2.8$) < pocket ($M = 4.6$) < hip ($M = 5.5$); by comparing the mean access times, we can see that the pocket condition yields a 66% longer access time than the wrist while the hip requires 98% more access time than the wrist.

For non-watch conditions, where the light sensor was used, *pocket time* was significantly affected by placement, but not mobility ($p = .128$), and there was a significant interaction between placement and mobility. There was no significant effect of placement for *hand time* ($M = 1.145$, $SD = .628$), but there was a significant effect for mobility as well as an interaction.

Some, although not all, of the other measures described earlier were significant (refer to Figure 14). The *answer time* was found to be significantly affected by mobility, but not placement (walking: $M = 1.227$, $SD = .304$; standing: $M = 1.479$, $SD = .583$). Measures non-significantly impacted by placement or mobility were *slide time* ($M = .071$, $SD = .72$) and *replacement time* ($M = 3.195$, $SD = 1.787$).

3.4 Quickdraw Discussion

The watch placement condition resulted in much faster access to the device. This finding is unsurprising, because participants did not have to remove the phone from the pocket or holster in order to use it.

In Figure 14, *access time* (from 2 to 4 in the timeline) is divided into two segments for non-wrist conditions: *pocket time* (from 2 to 3) and *hand time* (from 3 to 4). The statistics in Table 2 reveal that the majority of access time is consumed by pocket time[2]: on average, 78% of the time from the alarm until the participant started moving the slider was involved in getting the device out of the pocket or holster!

The watch condition also resulted in much more *consistent* access time to the device; the standard deviation of access time for the pocket and hip conditions is 2.5 and 2.9 times more than the wrist condition, respectively. The long time required to retrieve the phone from its holder may help explain why the participants in the Cui *et al.* study reported missing phone calls due to how they carried their phones [18].

While we anticipated the superior performance of the watch condition, we did not expect the holster to perform *worse* than the pocket condition for access time. Reviewing user comments made during the study, however, it becomes clear that poor holster design may account for this result. Despite our use of the manufacturer's holster, several participants complained during practice trials that the holster was difficult to use, and that difficulty may have persisted through the course of the study. An additional possibility is familiarity; while participants were presumably well-practiced with putting items into and removing them from the pockets of their own pants, none of the participants reported using a holster to carry their phone (Table 1) and therefore may have been slower with the holster than would have otherwise been expected. Given these factors, our results for the hip condition should probably be viewed as a worst case for holster access.

Finally, it was interesting to find that, in contrast to previous work [99], walking did not significantly impede user performance relative to standing still; our data was trending to show that standing resulted in slower access. One possible explanation for this difference is our inability to separate reaction time and access time with our current experimental design. It is possible that the walking conditions kept the participants more engaged in the experiment relative to the standing conditions, and therefore the reaction time was slower while standing.

[2]Note that in Table 2 $pocket + hand \neq access$. Two participants had light sensor problems, making hand and pocket time impossible to recover. Thus, these subjects were not included in this subsection of the table, making the quantities not exactly sum.

3.5 Conclusion

This section presented the Quickdraw study, which shows the wrist to be an ideal place—with respect to access time—to locate interactive devices. By placing a device on the wrist, the barrier to reaching a device fast enough to have the interaction be “micro” is removed. The next chapter investigates one possibility for wrist-based interaction.

CHAPTER IV

TOUCHSCREEN WRISTWATCH MICROINTERACTIONS

This chapter is adapted from a paper presented at the ACM International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI) [5]. The paper was co-authored by myself, Kent Lyons and Thad Starner.

The study presented here investigates interaction with a round-faced touchscreen wristwatch. As demonstrated in Chapter 3, the wrist is an ideal location for devices one wishes to quickly access; the question then becomes how to interact with a wrist-worn device. This chapter posits a traditionally round-faced watch that is touch sensitive, and investigates—at a low level—how one might interact with it. The research presented herein is important for several reasons:

- The introduction of new interaction concepts for the wrist. While circular interfaces and touchscreen watches are not new ideas (Section 2.2), the marriage of the two appears to be a novel concept. This chapter introduces the idea and proposes different application ideas that might take advantage of such an input/output device.
- Interaction techniques for a wrist-mounted round-faced touchscreen display. Due to its small size, round shape, and mounting location, traditional touchscreen techniques might not be appropriate for round touchscreen watches. This chapter discusses three possible touch-based techniques for interaction with such a device.
- Development of an error model for interaction techniques. On a display as small as that of a wristwatch, there is a tradeoff between ease of interaction and versatility. That is, if buttons are large, fewer of them may fit on the display. In this chapter I present a study into three interaction techniques that determines that optimal button size for each; in addition, I derive a generalized error model that allows for the calculation of error rate, number of buttons, or button thickness given any two of these

quantities.

4.1 Motivation

Research has shown that the amount of time required to access a device has a strong influence on whether a user will actually use that device at all [18, 93]. Chapter 3 demonstrated that the wrist is an excellent location to place devices that need to be accessed quickly. The question that is thus far left unanswered is how to create a usable interface on the wrist. In this chapter, we consider touchscreen watch interfaces.

Touch is an intuitive interaction method, and in the past decade touchscreens have become very popular for mobile devices. The number of commercially-released touchscreen wristwatches, however, has been very small, and they have in general not been successful. We believe that there are several reasons for this, two of which we address in this research.

The first issue with extant touchscreen watches is button size. All commercially released touchscreen watches thus far have shipped with a tiny stylus to allow the user to manipulate the onscreen interface, which invariably has minuscule buttons. Users, however, prefer to use their fingers to manipulate touchscreens, to avoid the time needed to retrieve the stylus [100].

A second issue with current touchscreen watches is lack of style. Wristwatches are often worn for fashion, but technological wristwatches appear to seldom fulfill this need. One reason may be that round wristwatches are preferred over rectangular ones. Indeed, many digital watches still have round faces surrounding the blocky digital time readout.

To address the issues of finger-usability and style inherent in current touchscreen wristwatches, we are investigating potential interfaces for touchscreen wristwatches with circular displays. We believe that a round-faced watch will be more accepted by consumers, and by designing the interface to be used by fingers—rather than by a stylus—we can make the watch easier to use.

In this chapter, we will present our preliminary findings on interaction with a round touchscreen watch. Inspired by non-wristwatch circular interfaces such as Pie Menus [14] and earPod [109], we consider the effects of placing wedge-shaped buttons around the perimeter of a circular screen. Most circular interfaces in the literature have been implemented for screens PDA-sized or larger, where the size of the buttons is less of a concern. On a wristwatch-sized screen, however, there will be a tradeoff between the number of buttons that can fit around the edge and the usability of the device; additionally, there may be a tradeoff between usability and the amount of non-button area left in the center for display purposes. Figures 15 and 16 show an example of the kind of application we hope to enable with this research.

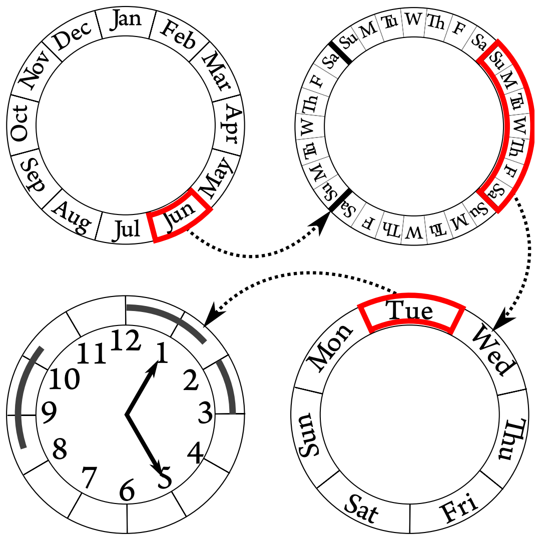


Figure 15: Mockup of a potential zooming interface for a calendar, using the *rim* style of interaction. The user selects the area denoted with the thick red line and "zooms" to the next level of detail with a sweep of the finger around the bezel. Each circle is printed at the actual size of the display used in the experiment.

Like rectangular screens, circular screens will necessarily have a bezel in order to accommodate the screen's electronics. As with the work by Froehlich *et al.* [23] and the work by Blaskó [25] we investigate the possibility of using the bezel to help guide the user's finger. We chose to study three types of finger/screen interactions that might be used on a round touchscreen watch: 1) tapping, as with a standard PDA-like application; 2) sliding in a straight line; or 3) sliding along the rim of the watch, using the bezel as a guide, as in [23] and [25].

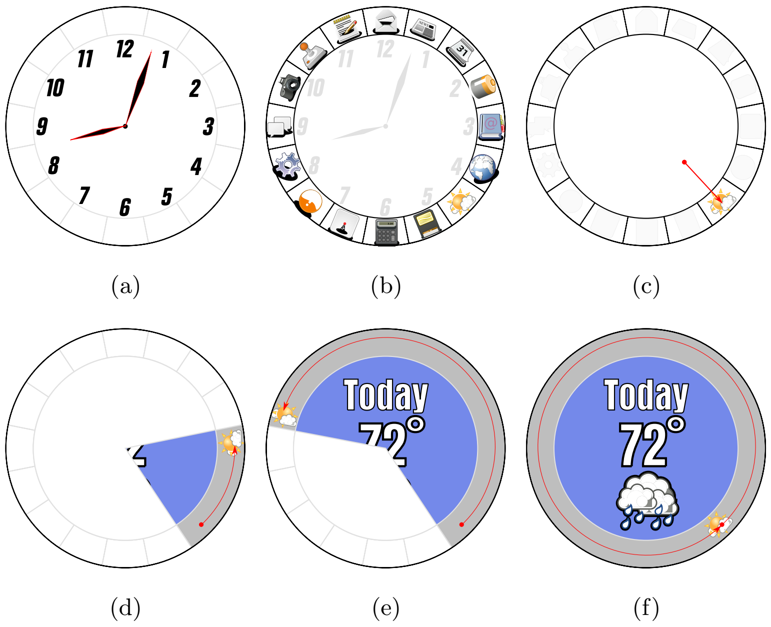


Figure 16: Proposed “card dragging” interaction using icons on rim of watch. (Interaction inspired by Blaskó’s card dragging technique [7].) (a): view of watch with no applications active; (b): user touches face to show application icons; (c): user selects desired application icon; (d)–(f): user drags application “card” out of “stack”. Each circle is printed at the actual size of the display used in the experiment.

4.2 Experiment

We conducted an experiment in order to investigate the size of buttons necessary to allow finger-based interaction on a small, round touchscreen watch. In particular, we wanted to understand the tradeoffs between number of buttons, non-button area remaining in the center, and error rate for each of the the three types of movement we discussed above.

4.2.1 Participants

Fifteen volunteers were recruited to participate in the experiment. Fourteen volunteers were male and one was female; ages ranged from 20 to 28. Participants were paid US $10 per hour for participation; no participant took more than one hour to complete the experiment. All participants were either right-handed or primarily used their right hand for control tasks such as mouse movement. Eight participants reported normally wearing a watch, and each stated they normally wore it on their left wrist.

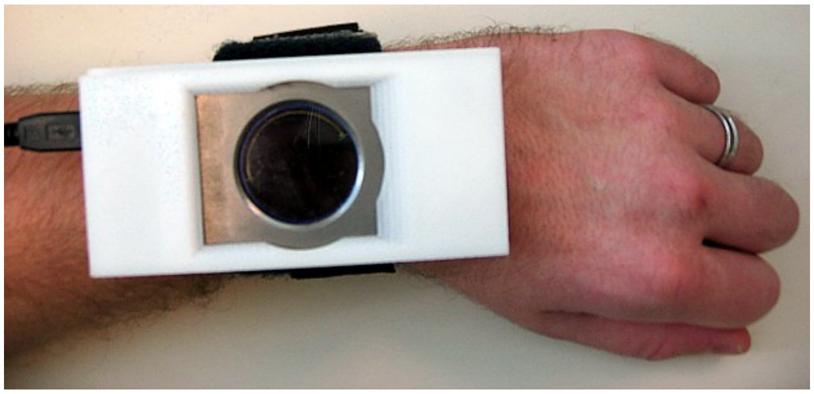


Figure 17: The simulated touchscreen watch.

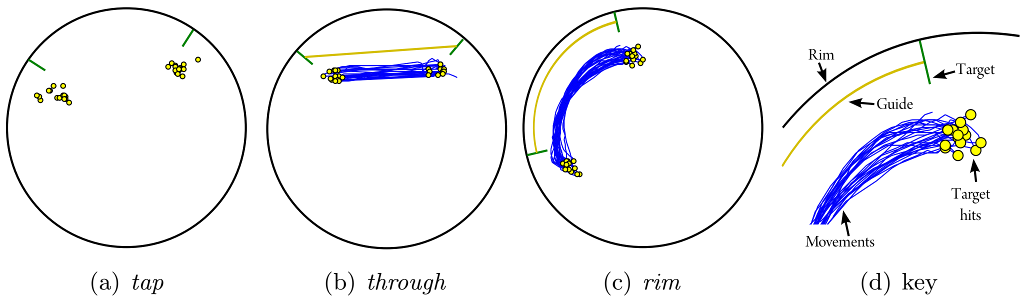


Figure 18: Targets with completed conditions for one participant. Rim, target and guide were displayed to participant as shown in (d); movements and target hits were displayed only to researcher. Images (a)–(c) are printed at the actual size of the display used in the experiment.

4.2.2 Variables

Movement type is our primary independent variable. The simplest method is using a tap to select, much like using a mouse to click on icons. This is the *tap* condition. Another method we investigated is sliding the finger in a straight line from one place on the surface to another; this forms the *through* condition. The final option is sliding the finger using the bezel as a guide, avoiding the center region of the watch. This is the *rim* condition.

The main dependent variables of interest are *button size* and *error rate*. Wobbrock *et al.*'s error model for Fitts' law [106] indicates that as button size decreases, error rate will increase; however, the bezel influences the movement of the user's finger, causing our task to be non-Fitts. Each button has both an *angular* and a *radial* width. As the buttons increase in angular width, fewer buttons will fit around the rim of the watch. As the radial width increases, less space will be available in the center of the watch for a display or other non-button purposes.

4.2.3 Apparatus

Lacking a programmable round touchscreen watch, we simulated one. We removed the case from a Motorola E680i touchscreen mobile phone and placed it into a custom-made plastic case (Figure 17). In order to simulate a round watch face with a bezel, we placed over the screen a custom-cut plate of 1.6mm (.06in) thick steel with a hole approximately 30mm (1.25in) in diameter, with a sloping bevel around the hole. The hole revealed an area of the phone's screen 200 pixels in diameter. The "watch" was mounted on each participant's arm using a velcro strap, with the visible portion of the screen centered on the arm approximately where a normal wristwatch face would be. We wrote software for the phone to present trials to participants and collect data.

4.2.4 Procedure

Our experiment is a 3 × 12 within-subjects factorial design. Participants performed a Fitts-style reciprocal 2D pointing task using their finger as the pointer. Two targets were displayed on the face of the wristwatch, and a secondary monitor instructed the participants to select the targets as quickly and accurately as possible, using the *tap*, *through* or *rim* movement type. For the *through* and *rim* conditions, a line was drawn on the watch face as a reminder of the motion to make (Figure 18). No feedback was given to participants during the trials, except to blank the screen to indicate the end of each trial.

4.2.5 Design

Each volunteer participated in 108 trials. The primary independent variables were movement type (*tap*, *through* and *rim*) and distance between targets (twelve distances were used, as explained below). To avoid learning effects, conditions were presented to participants in a balanced Latin square ordering.

In order to avoid biasing participants with any particular size of target, and following the literature [87], we selected targets as close to zero width as we could display. These were simple lines extending from the outside of the watch 18 pixels (2.7mm, .11in) towards the center (Figure 18). Each line was one pixel (.15mm) wide, or approximately 0.58° at the inner end of the target. While these targets are unrealistically small, they will allow us to consider the distribution of nearby hits.

To determine target positions, we divided the face of the watch into twelve (hour-sized) segments of 30° each, and then further subdivided each segment in half to get a minimum distance between targets of 15°. At the inner end of the targets, this is a about 21 pixels, or 3.2mm (.13in). We tested participants on all of the 15° increments between 15° and 180°, inclusive, for a total of twelve different distances. After choosing the distance between each pair of targets, the pair was centered at the rim of the watch on a location chosen from a uniform random distribution from 0–360°. This ensured that the data would cover the face of the watch.

Of note is that while each movement condition used the same set of target spacings, these distances differ between the *rim* and the other two conditions: a 180° distance in the *tap* or *through* condition will yield a 200 pixel (30mm) straight-line distance, but in the rim condition the participant's finger must travel in an arc, yielding a movement equal to πr, or 314 pixels (47.1mm). We chose to maintain equivalent target spacings between conditions to mimic the effects of an interface where relative button distances would be the same regardless of the method used to activate the buttons.

During each trial, the participant moved back and forth between the pair of targets 15 times, resulting in 30 movement end points. The end points in the *through* and *rim* conditions were determined by the user reversing movement direction; in the *tap* condition any tap on the screen was considered to be an end point. The software counted the end points and automatically ended the trial when the goal was reached. A block is a set of twelve trials (one per distance) with one movement type (*e.g.* *rim*); the blocks were repeated three times for each of the three movement types. Therefore we have 12 distances $\times$ 3 repetitions $\times$ 3 movement types = 108 trials per participant. In between each block, participants were given an enforced 30-second break to avoid fatigue; if desired, they could break for longer than 30 seconds.

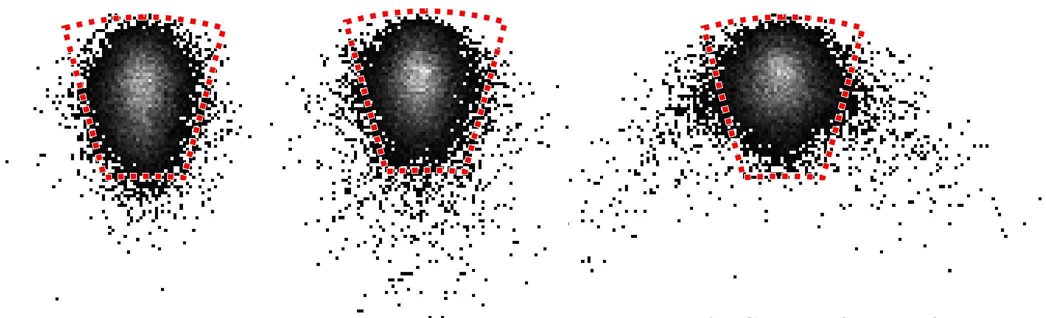


Figure 19: Distributions achieved by rotating clusters to 0°. Shown from left to right are points from *tap*, *through*, and *rim* conditions. Lighter colors denote a higher density of endpoints. Red dotted lines indicate example simulated button of 30° angular width and 50 pixels radial width.

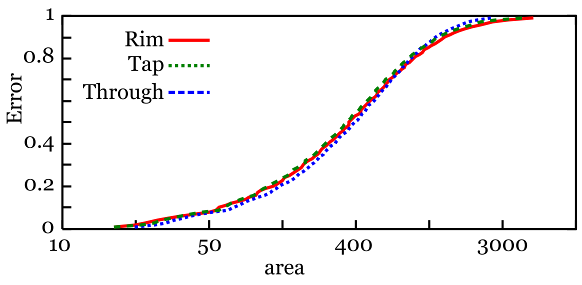


Figure 20: Button area (Equation 1) vs error rate. Note that x-axis is a log scale.

4.2.6 Results

After the experiment was complete, we processed and analyzed the data. Because we did not force participants to actually hit each target before proceeding to its opposite, it was sometimes difficult to determine which target the participant had been trying to hit, especially at small inter-target distances. Therefore, we used the k-means clustering algorithm to separate the points into two groups. As the seed means for the clusters, we used the innermost tips of the target lines. After clustering, the standard deviation (σ) of each cluster was calculated, and points falling greater than 3σ away from the cluster mean were discarded.

We next took the clusters from all of the participants and grouped them by movement type. We then rotated all of the clusters from each movement type so that the mean of each cluster was at 0°, giving an overall distribution for that movement type. The results are illustrated in Figure 19. We then calculated error rates for different radial and angular button widths. We accomplished this by creating simulated buttons (red dotted lines in

Table 3: Fits of error rate *vs* log of area data to Equation 2. The "all" condition represents the data for all three conditions aggregated together.

Condition	x	y	R^2	RMSE
tap	4.895	5.932	0.9939	0.0225
through	4.382	5.831	0.9964	0.0172
rim	4.247	5.878	0.9970	0.0157
all	4.476	5.881	0.9939	0.0224

Figure 19) for every integer combination of 0–100 pixels of radial width and 0–100° of angular width. Each button was placed against the rim and centered on 0°. The error rate for that button was then calculated simply by counting the number of inflection points falling inside the button versus those falling outside. For each combination of radial and angular widths, we also calculated the area (in pixels) of the button:

$$area = \frac{ang}{360}\pi\left(R^2 - (R - rad)^2\right) \tag{1}$$

where R is the radius of the watch (100 pixels), ang is the angular width of the button, and rad is the radial width. This area is equivalently expressible in terms of the number of buttons placed around the edge $numbt$ and the percentage of center area (relative to the entire surface area) available for other purposes $pctctr$:

$$area = \frac{\pi R^2}{numbt}(1 - pctctr)$$

By plotting the error rate (0–100%) against the log of the area of the button that resulted in that error rate, we observe a sigmoidal shape (Figure 20). We performed a least-squares fit to the data using the equation $1/\left(1 + x^{y-\ln(area)}\right)$ (2)

where $area$ is the button area as calculated in Equation 1. Close fits were found, as reported in Table 3.

The close fit of the data to Equation 2 allows us to model error rates for given angular and radial widths. A visualization of predicted error rates for one to fifty buttons with sizes such that zero to one-hundred percent of the center area remains is displayed in Figure 21, with contour lines at 10% error increments.

Because of finger width, participants could not touch the screen at the full radius; for the *tap* condition, the average distance away from the rim was 23.8 pixels (SD = 6.8),

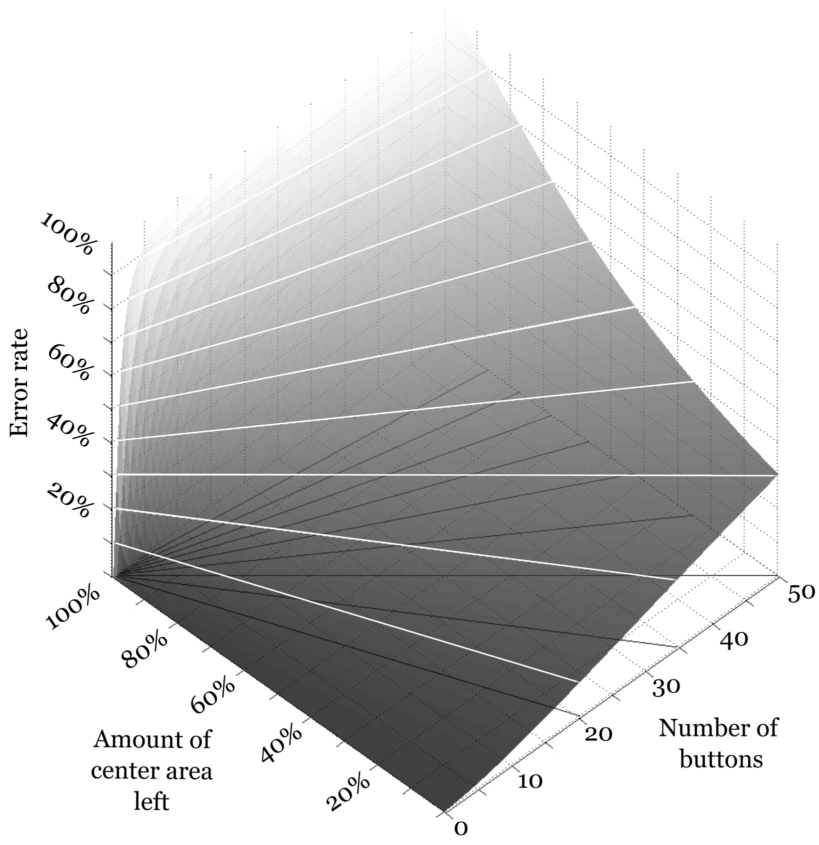


Figure 21: Number of buttons to be placed around the edge *vs* amount of surface area to be left for center area *vs* calculated error rate (Equation 2) using "all" row of Table 3. Error rate contour lines are displayed on the surface at 10% increments (solid white lines) and projected downwards (solid black lines).

for *through* 21.1 pixels (SD = 7.2), and for *rim* 17.7 pixels (SD = 6.7). We measured the amount of time for participants to move between targets; perhaps because of the small distances involved, there was no correlation between inter-target movement time and distance. Mean per-condition movement times are *tap*: 0.32s (SD = 0.08); *through*: 0.27s (SD = 0.14); *rim*: 0.39s (SD = 0.19).

4.2.7 Discussion

Given the equations above, we can now think about how a wristwatch interface might look. Because participants were requested to move between the targets as quickly and accurately as possible, the error rates can be assumed to be worst-case. By taking into account the desired application or situation, the correct number of buttons and center area can be found for a desired error rate. For example, if we want an error rate of 5% for ten buttons, we

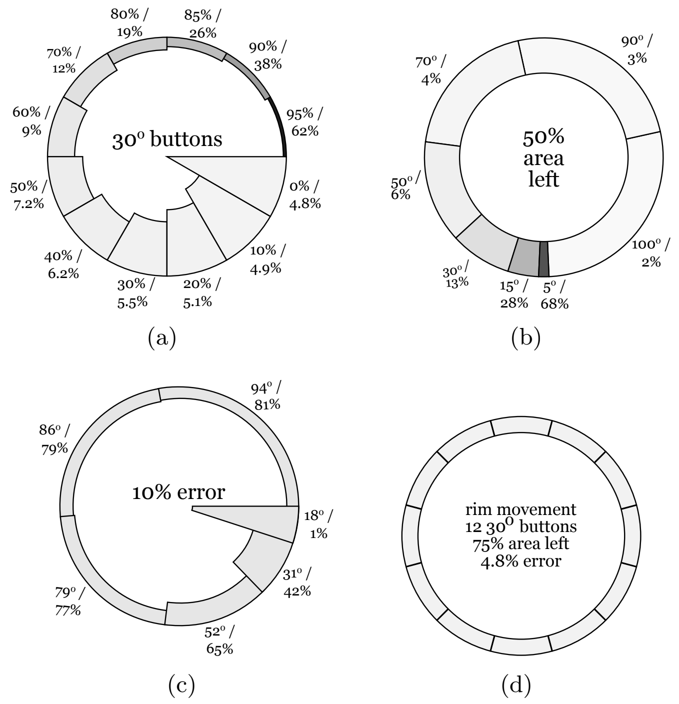


Figure 22: Illustrations of the effect of holding constant one of radial width, angular width, or error rate. Images (a)–(c) use x and y from the "all" row in Table 3. Image (a) shows the effect of holding angular width constant to 30° (for 12 buttons); each button is annotated with (the percentage of center area left / predicted error rate). Image (b) shows the effect of holding the radial width constant such that there is 50% center area remaining; each button is annotated with (the angular width of the resultant button / predicted error rate). Image (c) shows the effect of holding the error rate constant while picking a variety of radial and angular widths; each button is annotated with (angular width / percent of center area left). Finally, image (d) uses x and y from the "rim" row of Table 3 to illustrate a possible layout using 12 buttons with 75% center area left. Each circle is printed at the actual size of the watch face used in the experiment.

will have about 90% of the surface area left for the display. Figure 22 illustrates several possibilities for different combinations of button size and error rate based on our results.

As a general rule, it was difficult for participants to get their fingers close to the rim; this is reflected in the sharp slope nearing 100% in the "center area" axis of Figure 21. We would therefore recommend keeping the radial width of buttons above the mean distance from the edge for each movement type.

4.3 Conclusions

The results of our experiment yielded a mathematical model of error rate given the angular and radial widths for buttons placed around the edge of a circular touchscreen watch. We determined constants x and y for the model for three inter-target movement types: tapping (*tap*), sliding in a straight line between targets (*through*) and sliding along the bezel of the display (*rim*).

Our experiment gave us a wealth of data and many directions for future exploration. We wish to experimentally validate our results by running another study with various button sizes to ascertain how well our model predicts error rate with real buttons, rather than simple lines.

We intend to investigate whether there is a bias in error rate based on the location of the target on the watch: observation during the experiment suggested that targets in the upper-left quadrant of the watch—from 9 o'clock to 12 o'clock—are more likely to be obscured by the user's finger, while targets on the bottom rim—from 4 o'clock to 8 o'clock—may be more difficult to hit due to the shape of the user's finger.

Other work has addressed the issue of screens obscured by fingers, for example the Shift offset cursor technique by Vogel and Baudisch [100]; however, such techniques usually rely on having extra screen real-estate where the obscured content can be displayed. Possible methods for accomplishing a similar interaction style in our extremely limited display space include mirroring the content across the face of the watch or using fisheye views.

The ultimate motivation for this work is to allow fast access and use of the watch in any situation, including while actually in motion. We are particularly interested in whether the stabilization offered by the bezel in the *rim* movement condition confers an advantage over the other two movement types while the user is in motion.

The next chapter considers an alternate method for enabling microinteractions: through gesture.

CHAPTER V

MAGIC: A GESTURE DESIGN TOOL

In this Chapter, I discuss a second way of enabling mobile microinteractions: through gesture. I have created a Multiple Action Gesture Interface Creation tool (MAGIC) to assist designers of motion gesture-based interfaces. Gesture design can be a complex task, and there are few extant tools that support it. In this chapter, I present the design of the MAGIC software. I discuss the rationale for the design, the workflow MAGIC is designed to support, and the back- and frontend components of the design. In Chapter 6, I present the results of an evaluation of MAGIC, and in Chapter 7 I discuss the implications of the evaluation results with respect to future designs.

The design and evaluation of MAGIC are important for several reasons:

- *Motion gesture design software for non-expert users* is presented. Even those familiar with statistical machine learning can have difficulty using it in software [71], and for non-experts the difficulties are much worse. MAGIC is a tool to assist users in designing motion gestures that simplifies much of the complexity inherent in such a task.

- *A method for designing low-false-positive gestures* is included in MAGIC. Even after successfully creating a set of gestures, a designer must ensure that they are only activated when the user desires it. MAGIC includes a feature called the *Everyday Gesture Library* which assists gesture designers in designing freeform gestures that are unlikely to be falsely triggered by a person's everyday movements.

- *Strategies used by non-domain experts in designing motion gestures* are detailed. These strategies could be used in the future as automated suggestions to help other non-experts design gestures that may work well.

- *A corpus of everyday movements by seven users of varying backgrounds.* Although due to privacy concerns and IRB regulations the accompanying video cannot be released, I have an extensive corpus of accelerometer recordings extending for over 60 hours. This data may be used for future gesture design and recognition testing using MAGIC or independently.

5.1 Motivation

As addressed in the Introduction (Chapter 1), this dissertation deals with microinteractions. In the case of instant commands, an interaction so fast that it is almost thought-free is desirable, and gestures may fit this requirement. However, as discussed in Related Work (Chapter 2), motion gestures (as opposed to pen gestures) have not become prevalent outside of gaming systems such as the Nintendo Wii. Research indicates that users prefer devices that are fast to access [18, 94], and motion gestures can provide this desired speed by obviating the need to press buttons or look at screens. Gestures can even enable hands-free usage for on-body devices such as wristwatches. Users could control tiny music players with subtle shoulder movements, look at upcoming appointments on a watch display without having to touch the watch, or dial a mobile phone with a wave of the hand. Both the sensing and gesture recognition technology exist to create these kinds of interfaces, but currently the Apple iPod is the only device implementing quick gestures, with its "shake to shuffle" motion. I see two causes for this lack of gesture recognition in everyday life.

The first issue is that interaction designers are not generally domain experts in gesture or pattern recognition [22]. A number of off-the-shelf tools for experimenting with pattern recognition exist, such as Weka [105]; these tools, however, act more as libraries of techniques rather than full-fledged design tools.

The second issue is that testing gestures in everyday life can be very difficult. This challenge is not just normal user testing; for gestures to really move into everyday usage, they must be usable in everyday situations. In particular, only movements that are intended for the device should cause functionality to be activated; that is, performing normal activities such as eating, walking or normally gesticulating during conversation should not cause unwanted activity on the device. One solution to this problem is a "push to gesture" mechanism, where a user first presses a button and then makes a gesture; however, such an interaction obviates the need for gestures in the first place, as the user could simply press the button to activate the desired functionality without making a gesture. Pushing to gesture can also slow down the interaction and make it inconvenient for the user.

With these needs in mind in this chapter, I define a list of desiderata for a gesture design tool and introduce my system for Multiple Action Gesture Interface Creation (MAGIC), and describe its implementation. In Chapter 6 I discuss the results of an evaluation of MAGIC's usability, and in Chapter 7 I discuss future work and implications for design of further systems.

5.2 Motion gesture design tool desiderata

In my experience, creating a gesture system has three basic stages (Figure 23). In the first stage (a), the designer gathers requirements, performs formative user studies and market research, and decides what functionality to consider controlling with gesture. In the second stage (b), the designer determines how motions by the user will map to functionality activated on the device. She ensures that only intended movements activate functionality, that the gestures work reliably, and performs initial user testing, especially related to how the designed gestures work in conjunction with a user's everyday movements. The last stage (c) is summative: the designer performs final user testing and deploys the finished product.

Figure 24 illustrates how—in the experience of my advisor's research group [3, 10, 40, 55, 91, 101, 102]—gesture interface design currently proceeds. If gestures conflict with each other, new gestures must be created. If gestures conflict with everyday life—that is, if there is a high rate of false positives—new gestures must be created. The process is highly linear, and it can be difficult to create a working gesture set.

In this chapter, I discuss the creation of a tool to support the middle stage of design. I have identified the following desiderata for such a tool; the tool should:

allow non-expert use. Just as desktop UI designers are not required to know details about circuit design, USB protocols, or operating system drivers to build a system that responds to mouse clicks, interaction designers should not be required to understand

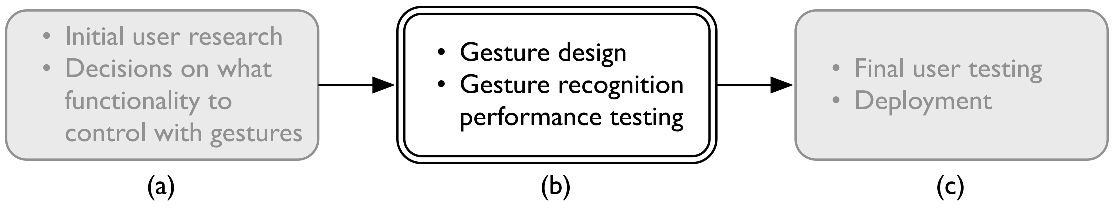

Initial user research
Decisions on what functionality to control with gestures
Gesture design
Gesture recognition performance testing
Final user testing
Deployment
(a)
(b)
(c)

Figure 23: The three stages of gesture design.

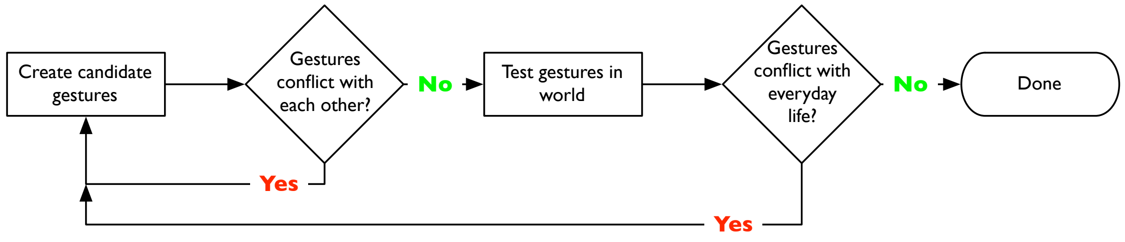

Create candidate gestures
Gestures conflict with each other?
No
Yes
Test gestures in world
Gestures conflict with everyday life?
No
Yes
Done

Figure 24: The current method used to design gestures—a linear process which must be restarted when problems arise.

the underlying complexities of gesture recognition in order to build a working system. This philosophy has been applied in other projects that seek to make machine learning, pattern recognition and computer vision accessible to non-expert designers [22, 30, 54, 59].

allow expert use. As designers use a system or a particular algorithm, they will naturally gain more expertise, and may wish to push beyond novice-level support. Expert use should be supported, and a smooth novice-to-expert transition should be possible.

encourage iteration. Iterative design is one of the cornerstones of good UI creation practices [66]. Tools should encourage iteration by making it easy to explore alternatives. In terms of gesture design, users should be able to quickly try different motions for a gesture-activated function and experiment with recognition parameters in order to get the desired results.

support retrospection. In contrast to pen gestures, motion gestures can be difficult to represent graphically, with only limited success in this area [43, 50]. However, designers still need the ability to reflect on the gestures they've created, understand

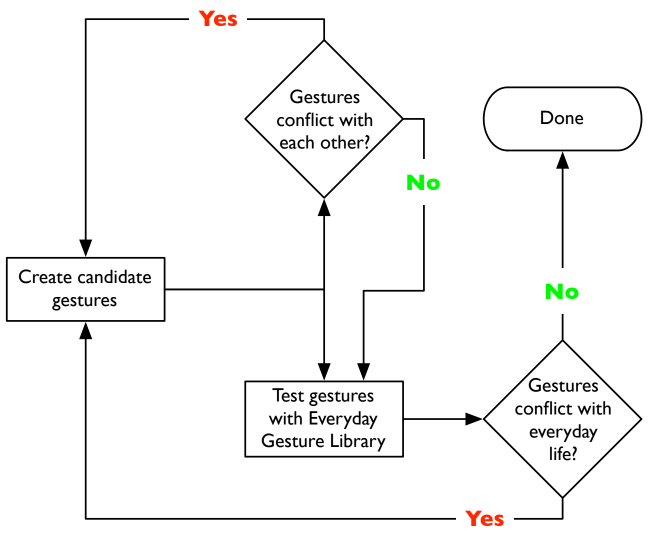

Yes
Gestures conflict with each other?
Done
No
Create candidate gestures
No
Test gestures with Everyday Gesture Library
Gestures conflict with everyday life?
Yes

Figure 25: The procedure used within MAGIC to design gestures. Testing whether gestures conflict with each other and if they might have high numbers of false positives in everyday life (tested via proxy with the EGL) are independent activities and may be done in any order.

how they are similar or different from other gestures, and to remember what motion corresponds to what function. I define the term *retrospection* to be a facility to allow the user to retrospectively consider input in as close to its original form as possible. MAGIC provides this functionality primarily through video recordings of the user performing gestures.

support further testing. Outside of general recognition concerns, there may be a number of other needs that the designer should consider. Included may be social acceptability, memorability, and usability in different situations. The designer may want to customize the gesture set for different user groups or cultures. Support should be available for the designer to perform further testing after the initial design phase.

I have developed MAGIC, a Multiple Action Gesture Interface Creation tool designed to meet these goals.

5.3 MAGIC: A Gesture Design Tool

MAGIC is partially inspired by the design/test/analyze model developed by Klemmer *et al.* [41], and supports a similar three-part workflow. The parts are not strictly linear; a designer will move between them as the gesture design process progresses. MAGIC reflects these stages with three tabs in the interface, between which the designer may freely move: *Gesture Creation*, *Gesture Testing*, and *Everyday Gesture Library* (see Figures 28, 29, and 30). The stages—illustrated in Figure 25—are:

1. **Gesture Creation** In this stage, the designer creates *gesture classes* and *gesture examples.* A gesture class represents one kind of movement—such as a punching forward motion—that usually maps to one function in the interface, such as "volume up". A gesture example is an instance of actual recorded motion data associated with a class. In MAGIC, the gesture examples are used directly in recognition; see section 5.5.1 for more information. Typically a designer will create several examples for each class in order to account for variation in how a user might make the gesture.

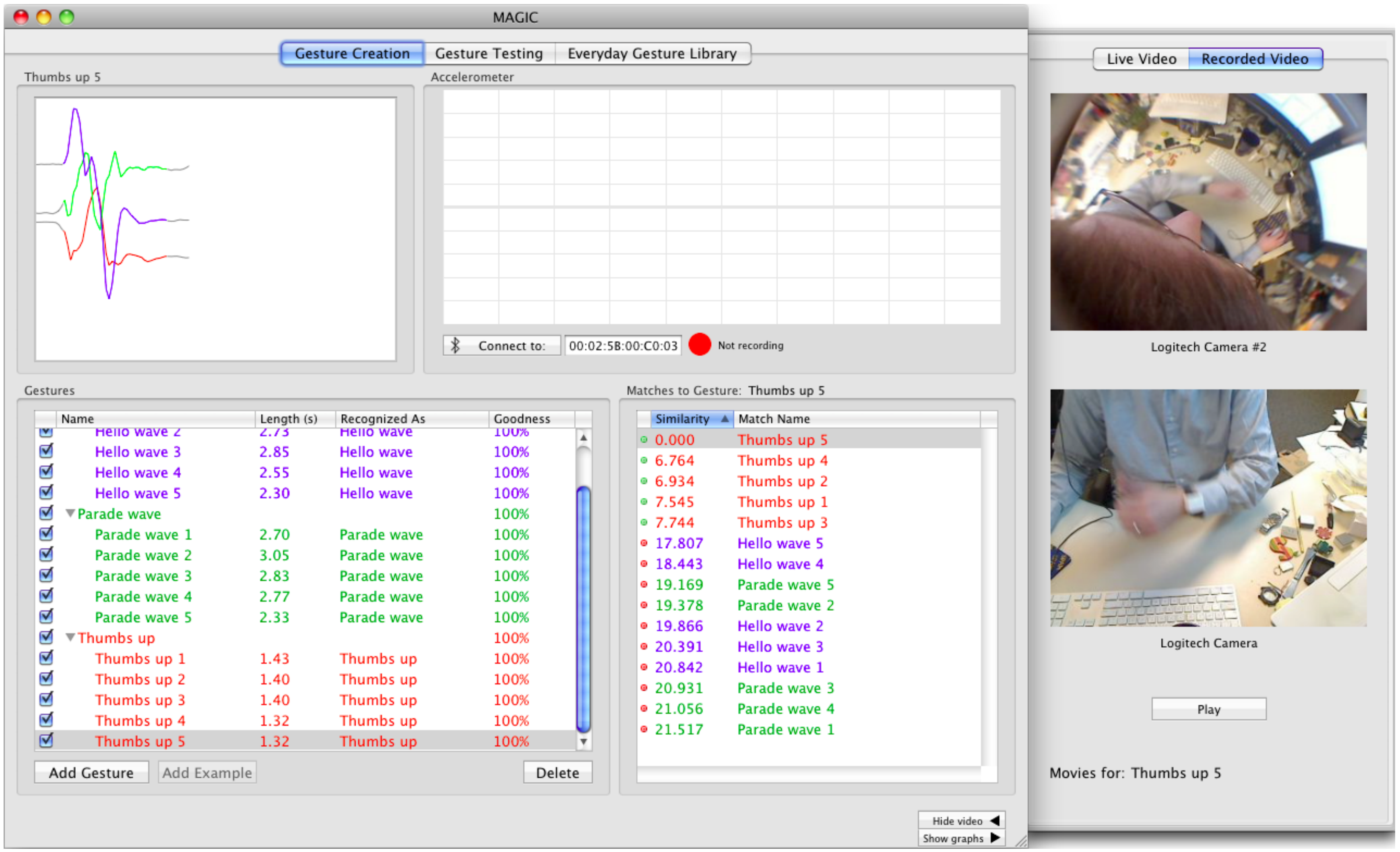
   The *Gesture Creation* tab (Figure 28) includes support for creating gesture classes, recording examples, and understanding how examples and classes relate to each other in terms of recognition performance. Gestures may be created and deleted, or, as an alternative to deletion, gestures and examples may be temporarily *disabled* by toggling the checkbox next to the item.

2. **Gesture Testing** In the testing phase, the designer tests recognition by making motions that should be recognized as one of the gesture classes trained in the creation phase, or by making motions that should *not* be recognized. For example, the designer might perform a punching forward motion to make sure it is recognized, and also reach for a glass of water to make sure that it is not falsely recognized as the punching motion.

   The *Gesture Testing* tab (Figure 29) allows the designer to create free-form sequences of movements that—from a recognition standpoint—are treated exactly the same as if they were performed live. The results of recognition are visualized, and the designer can re-run tests multiple times after adjusting parameters. If portions of a test are not recognized as members of the proper class, the user can select that portion of the test sample and add it to a class as a gesture example.

3. **False Positive Testing** One potential pitfall when creating gestures intended for everyday use is that end-users may perform actions that the designer can't anticipate, which might lead to unwanted activation of functionality. For example, the designer might think that adding a twist to the end of a forward punch will take care of any confusion between the gesture performed intentionally and picking up a glass of water, but it might instead be activated when turning a doorknob. For this reason, it is important to test the gestures during the actual daily activities of representative end-users.

   The *Everyday Gesture Library* tab (Figure 30) presents the designer with an interface similar to the Gesture Testing tab. In this case, however, the movements used for recognition are pre-recorded by a representative set of users, allowing the designer to determine if the system will confuse the created gestures with end-users' everyday movements. This pre-recorded data is called the "Everyday Gesture Library" (EGL).



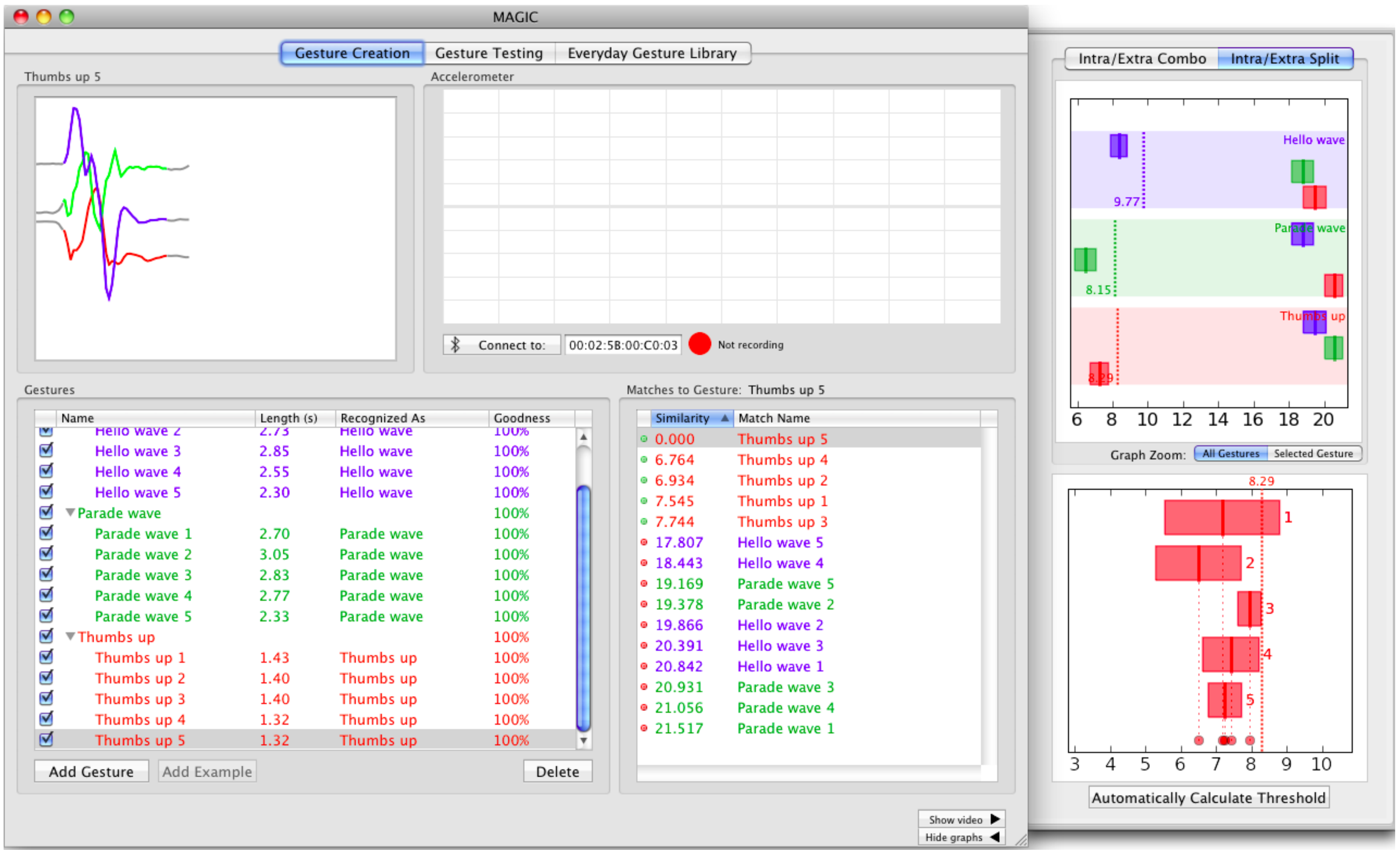
Gestures

Name	Length (s)	Recognized As	Goodness
Hello wave 2	2.73	Hello wave	100%
Hello wave 3	2.85	Hello wave	100%
Hello wave 4	2.55	Hello wave	100%
Hello wave 5	2.30	Hello wave	100%
▼ Parade wave			100%
Parade wave 1	2.70	Parade wave	100%
Parade wave 2	3.05	Parade wave	100%
Parade wave 3	2.83	Parade wave	100%
Parade wave 4	2.77	Parade wave	100%
Parade wave 5	2.33	Parade wave	100%
▼ Thumbs up			100%
Thumbs up 1	1.43	Thumbs up	100%
Thumbs up 2	1.40	Thumbs up	100%
Thumbs up 3	1.40	Thumbs up	100%
Thumbs up 4	1.32	Thumbs up	100%
Thumbs up 5	1.32	Thumbs up	100%

Matches to Gesture: Thumbs up 5

Similarity ▲	Match Name
0.000	Thumbs up 5
6.764	Thumbs up 4
6.934	Thumbs up 2
7.545	Thumbs up 1
7.744	Thumbs up 3
17.807	Hello wave 5
18.443	Hello wave 4
19.169	Parade wave 5
19.378	Parade wave 2
19.866	Hello wave 2
20.391	Hello wave 3
20.842	Hello wave 1
20.931	Parade wave 3
21.056	Parade wave 4
21.517	Parade wave 1

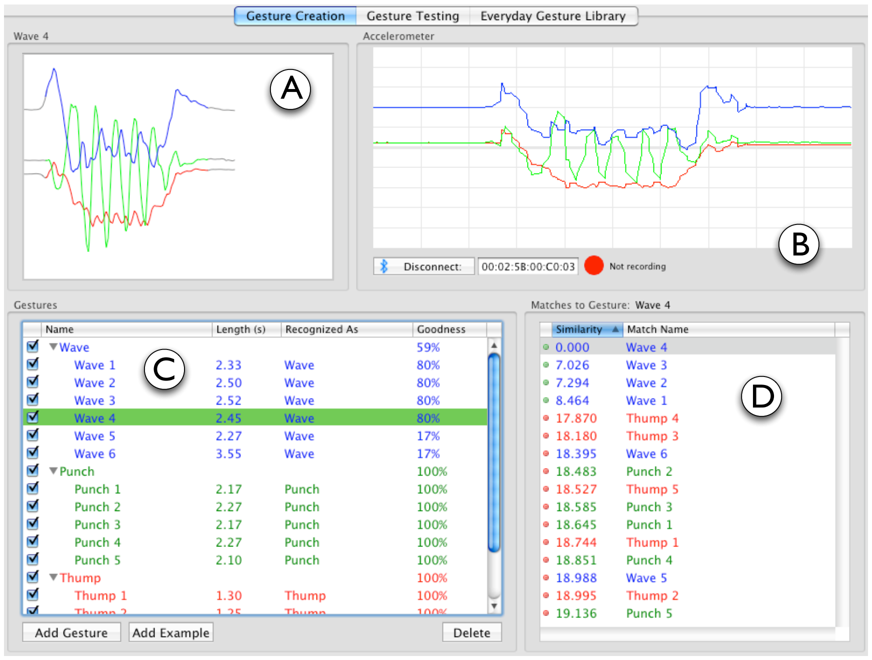
Figure 26: The MAGIC interface as seen by the user. The auxiliary window on the right (a “drawer” in Apple Macintosh parlance—as in “desk drawer” rather than “that which draws”) shows live and recorded video of a user performing gestures. The top video is from the hat-mounted camera and the bottom from the monitor camera (Figure 33).



Name	Length (s)	Recognized As	Goodness
Hello wave 2	2.73	Hello wave	100%
Hello wave 3	2.85	Hello wave	100%
Hello wave 4	2.55	Hello wave	100%
Hello wave 5	2.30	Hello wave	100%
Parade wave			100%
Parade wave 1	2.70	Parade wave	100%
Parade wave 2	3.05	Parade wave	100%
Parade wave 3	2.83	Parade wave	100%
Parade wave 4	2.77	Parade wave	100%
Parade wave 5	2.33	Parade wave	100%
Thumbs up			100%
Thumbs up 1	1.43	Thumbs up	100%
Thumbs up 2	1.40	Thumbs up	100%
Thumbs up 3	1.40	Thumbs up	100%
Thumbs up 4	1.32	Thumbs up	100%
Thumbs up 5	1.32	Thumbs up	100%

Similarity	Match Name
0.000	Thumbs up 5
6.764	Thumbs up 4
6.934	Thumbs up 2
7.545	Thumbs up 1
7.744	Thumbs up 3
17.807	Hello wave 5
18.443	Hello wave 4
19.169	Parade wave 5
19.378	Parade wave 2
19.866	Hello wave 2
20.391	Hello wave 3
20.842	Hello wave 1
20.931	Parade wave 3
21.056	Parade wave 4
21.517	Parade wave 1

Figure 27: The MAGIC interface as seen by the user. The auxiliary window on the right (a “drawer” in Mac parlance—as in “desk drawer” rather than “that which draws”) shows graphs illustrating how the gesture classes and examples in the system relate to each other.



Name	Length (s)	Recognized As	Goodness
▼Wave			59%
Wave 1	2.33	Wave	80%
Wave 2	2.50	Wave	80%
Wave 3	2.52	Wave	80%
Wave 4	2.45	Wave	80%
Wave 5	2.27	Wave	17%
Wave 6	3.55	Wave	17%
▼Punch			100%
Punch 1	2.17	Punch	100%
Punch 2	2.27	Punch	100%
Punch 3	2.17	Punch	100%
Punch 4	2.27	Punch	100%
Punch 5	2.10	Punch	100%
▼Thump			100%
Thump 1	1.30	Thump	100%
Thump 2	1.25	Thump	100%

Similarity	Match Name
0.000	Wave 4
7.026	Wave 3
7.294	Wave 2
8.464	Wave 1
17.870	Thump 4
18.180	Thump 3
18.395	Wave 6
18.483	Punch 2
18.527	Thump 5
18.585	Punch 3
18.645	Punch 1
18.744	Thump 1
18.851	Punch 4
18.988	Wave 5
18.995	Thump 2
19.136	Punch 5

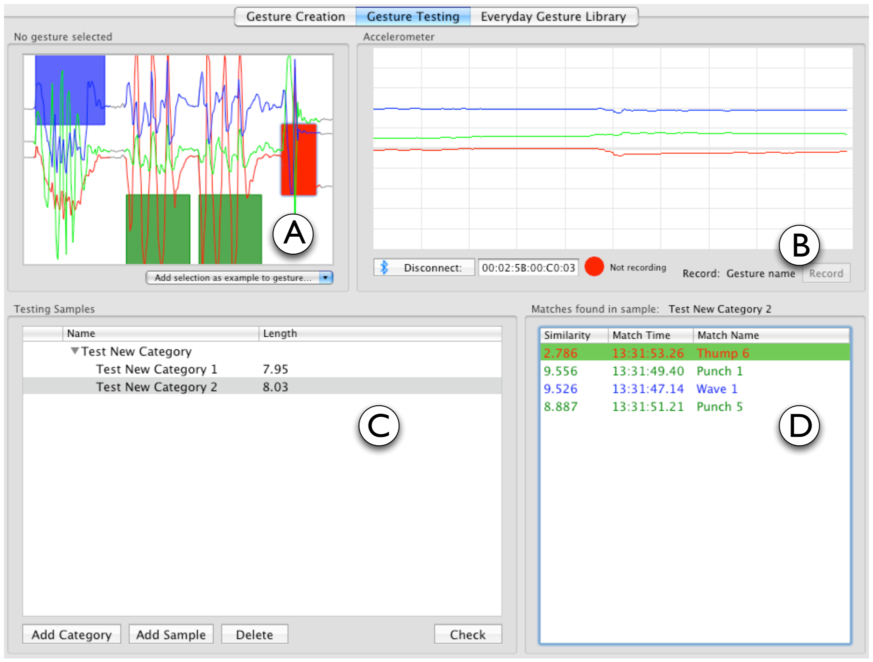
Figure 28: The Gesture Creation tab. A: recorded gesture view; B: live sensor view; C: list of gestures and gesture examples; D: sorted list of distances from currently selected example to every other example.

5.4 *Gesture Design Workflow Support*

Designing gestures can be a complex task, especially for those not familiar with pattern recognition techniques and terminology. MAGIC provides several features to assist users in their task; recall that the three stages are *gesture creation*, *gesture testing*, and *false positive testing*.

5.4.1 Everyday Gesture Library

In order to choose an effective set of gestures for everyday use, it is essential to test each iteration of the gestures with users: a user's natural motions may, to the computer, resemble the defined gesture, and therefore trigger undesired actions. My advisor's research group has over a decade's experience with gesture interfaces for everyday life [3, 10, 40, 55, 91, 101, 102] and has struggled with this issue. One solution is to lengthen the design cycle, and, each time a new gesture is designed, test it in the field. Doing so can lead to very long iterations, however; the designer of one interface spent two weeks in this manner before giving up and adding a push-to-gesture feature to the system.

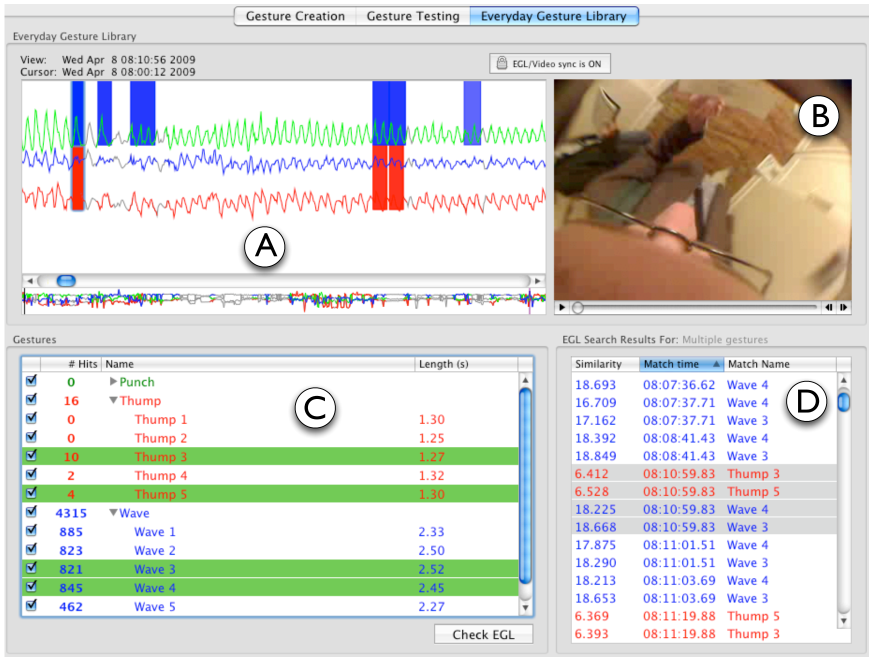


Matches found in sample: Test New Category 2

Name	Length
▼Test New Category	
Test New Category 1	7.95
Test New Category 2	8.03

Similarity	Match Time	Match Name
2.786	13:31:53.26	Thump 6
9.556	13:31:49.40	Punch 1
9.526	13:31:47.14	Wave 1
8.887	13:31:51.21	Punch 5

Figure 29: The Gesture Testing tab. A: recorded test sample view with boxes highlighting matches; B: live sensor view; C: list of test samples; D: list of gestures matching currently selected test sample.



View: Wed Apr 8 08:10:56 2009
Cursor: Wed Apr 8 08:00:12 2009

	# Hits	Name	Length (s)
☑	0	▶Punch	
☑	16	▼Thump	
☑	0	Thump 1	1.30
☑	0	Thump 2	1.25
☑	10	Thump 3	1.27
☑	2	Thump 4	1.32
☑	4	Thump 5	1.30
☑	4315	▼Wave	
☑	885	Wave 1	2.33
☑	823	Wave 2	2.50
☑	821	Wave 3	2.52
☑	845	Wave 4	2.45
☑	462	Wave 5	2.27

EGL Search Results For: Multiple gestures

Similarity	Match time ▲	Match Name
18.693	08:07:36.62	Wave 4
16.709	08:07:37.71	Wave 4
17.162	08:07:37.71	Wave 3
18.392	08:08:41.43	Wave 4
18.849	08:08:41.43	Wave 3
6.412	08:10:59.83	Thump 3
6.528	08:10:59.83	Thump 5
18.225	08:10:59.83	Wave 4
18.668	08:10:59.83	Wave 3
17.875	08:11:01.51	Wave 4
18.290	08:11:01.51	Wave 3
18.213	08:11:03.69	Wave 4
18.653	08:11:03.69	Wave 3
6.369	08:11:19.88	Thump 5
6.393	08:11:19.88	Thump 3

Figure 30: The Everyday Gesture Library tab. A: EGL view with boxes highlighting gesture occurrences; B: EGL video synchronized to EGL view; C: gestures with number of occurrences in EGL (# Hits); D: list of occurrences in EGL for selected gestures.

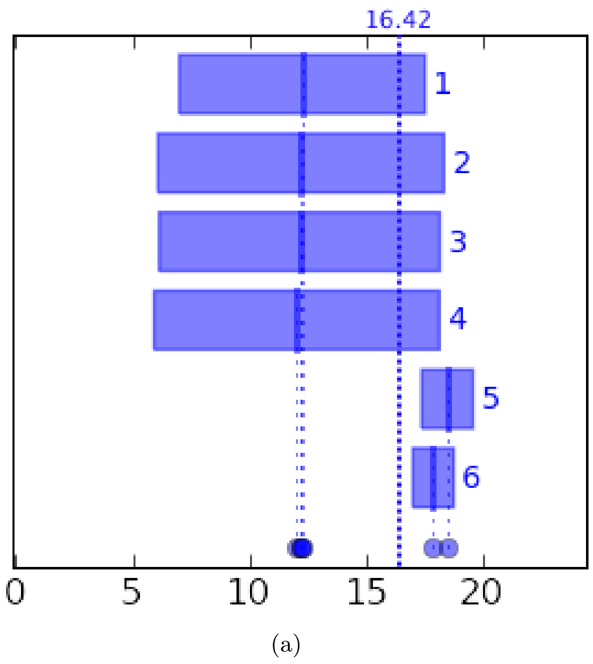
This issue can be particularly severe when designing a variety of highly iconic gestures. As I discovered in my evaluation of MAGIC (Chapter 6), one common tactic in gesture design is to make related gestures map to related commands; for example, moving one's hand upward for *Volume Up* and downward for *Volume Down*. Interrupting this process to test gestures makes for slow work, and if the designer discovers that, for example, *Volume Down* is not a good gesture, it may mean redesigning *Volume Up* as well.

MAGIC offers a partial solution to this problem. Rather than requiring user testing in the initial phases of gesture design, MAGIC allows designers to rapidly iterate through designs while increasing the likelihood that the chosen gestures will not be confused with everyday movements. MAGIC uses a corpus of pre-recorded data that is representative of the everyday motions of the designer's target population. The designer recruits a number of people to wear the same sensor that will be used in the end product. Those people then perform their daily activities, not explicitly interacting with the sensor at all, but simply allowing it to record data. The recorded data set—called an "Everyday Gesture Library" (EGL)—is used by MAGIC to help the designer more rapidly iterate through gestures. At any point in the workflow, the designer may test the currently defined set of gestures against the EGL to see how frequently the gestures occur. Each occurrence indicates a time when the end user would have accidentally triggered the functionality represented by the gesture. If the number of occurrences for a gesture are deemed unacceptable by the designer, she may continue to iterate.

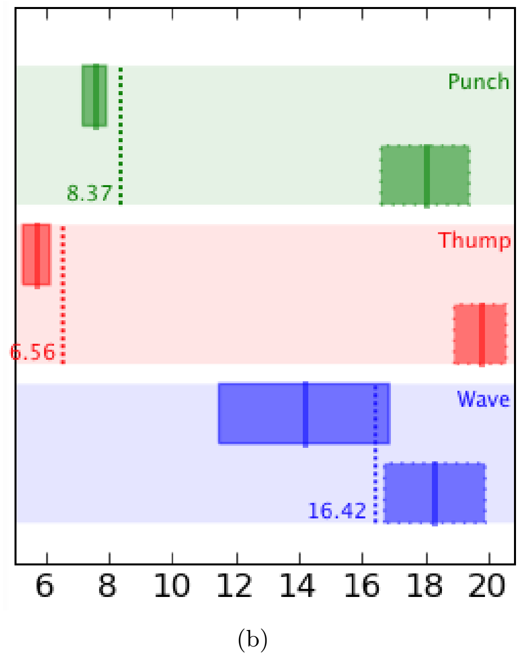
5.4.2 Retrospection

One of the most important aspects of MAGIC is *retrospection*—the ability to return to previously-created content and review the actions taken. MAGIC implements retrospection by graphically plotting recorded gestures and by making available video of the designer performing the gesture (Figure 26).

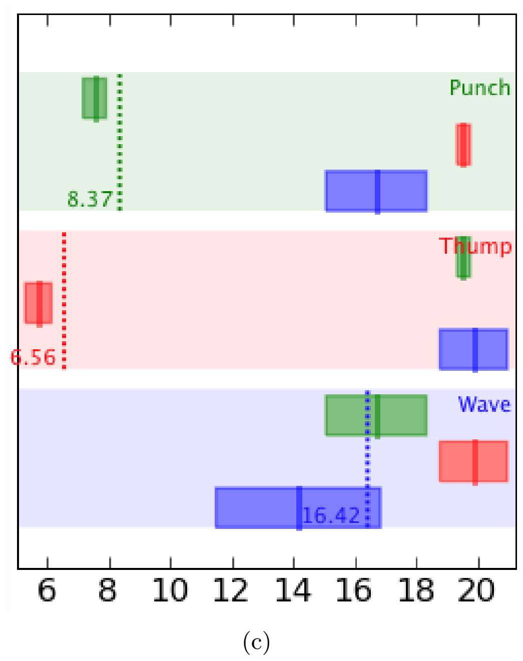
In both the creation and testing phases, the designer records motions. MAGIC displays a continuously-updating graph of the output from the accelerometer (Figure 28B). The x, y, and z axes are each displayed as a red, green, or blue line, respectively. This visual style



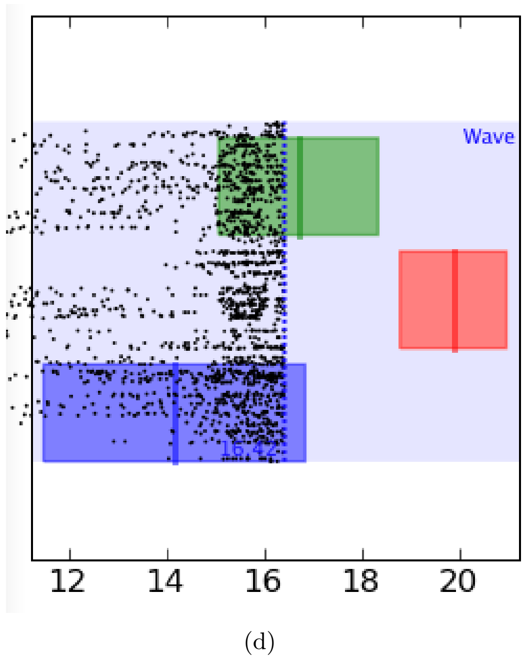
(a)



(b)



(c)



(d)

Figure 31: Mean and standard deviation distance graphs for (a) each example within a class, (b) each class versus all other classes in aggregate, (c) each class versus all other classes individually, and (d) a magnified view of a single row of (c). The horizontal scale in each case represents the DTW distance. The dots in (d) represent occurrences in the EGL; see Section 5.4.4 for further explanation.

is maintained after a gesture has been recorded: each training example is displayed in the same way (Figure 28A). The live output is located to the right of the training example display and continually scrolls to the left, to give the impression that recorded training examples have simply continued leftward to be "captured" by the example's display.

When experimenting with many potential movements for different functions, it can be easy to forget what movements were made. MAGIC automatically records video of the designer's movements during gesture creation and testing sample creation. Two cameras with 170° fisheye lenses are used: one is mounted on top of the designer's monitor, and the other is located in the brim of a hat (Figure 33). The hat-mounted camera provides a first-person view of movements to the designer, while the monitor-mounted camera offers a view that may be more legible to others.

As the accelerometer view and the video are representations of the same thing—a movement by the designer—they change in concert. The user can scrub back and forth through the video by dragging a cursor within the recorded accelerometer display. The user may also play back video at any time, in which case the cursor follows along with the video.

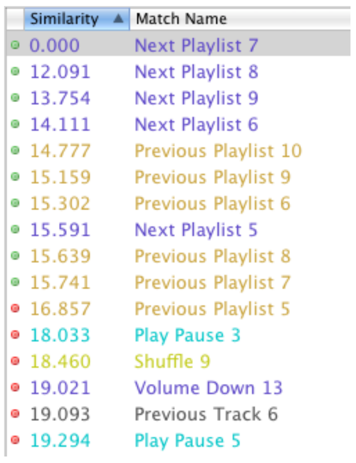
5.4.3 Visualization: Gesture Design

Table 4: Confusion matrix between eight classes. The labels across the top represent the class an example belongs to, while the labels along the side represent the class that each example was classified as. It illustrates, for example, that class *B* was incorrectly classified as *F* three times and as *D* once.

	A	B	C	D	E	F	G	H
A	10	0	0	0	0	0	0	0
B	0	6	0	0	0	0	0	0
C	0	0	10	0	0	0	0	0
D	0	1	0	10	0	0	0	0
E	0	0	0	0	10	0	0	0
F	0	3	0	0	0	10	0	0
G	0	0	0	0	0	0	10	0
H	0	0	0	0	0	0	0	10

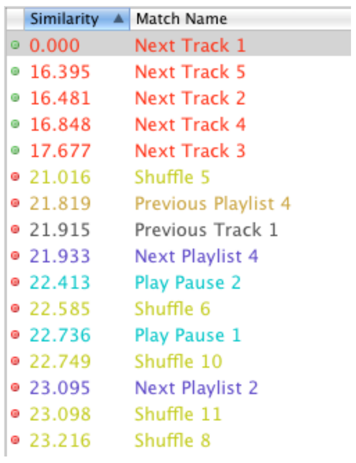
For classification tasks such as gesture recognition, a *confusion matrix* is a standard visualization of classification results, helping the user understand the source of incorrect classifications. An example confusion matrix is shown in Table 4. Because confusion matrices can be difficult to read for non-experts [54], MAGIC provides different visualizations derived from the same information.

The confusion matrix in Table 4 was constructed by comparing every example in every



	Similarity ▲	Match Name
🟢	0.000	Next Playlist 7
🟢	12.091	Next Playlist 8
🟢	13.754	Next Playlist 9
🟢	14.111	Next Playlist 6
🟢	14.777	Previous Playlist 10
🟢	15.159	Previous Playlist 9
🟢	15.302	Previous Playlist 6
🟢	15.591	Next Playlist 5
🟢	15.639	Previous Playlist 8
🟢	15.741	Previous Playlist 7
🔴	16.857	Previous Playlist 5
🔴	18.033	Play Pause 3
🔴	18.460	Shuffle 9
🔴	19.021	Volume Down 13
🔴	19.093	Previous Track 6
🔴	19.294	Play Pause 5

(a)



	Similarity ▲	Match Name
🟢	0.000	Next Track 1
🟢	16.395	Next Track 5
🟢	16.481	Next Track 2
🟢	16.848	Next Track 4
🟢	17.677	Next Track 3
🔴	21.016	Shuffle 5
🔴	21.819	Previous Playlist 4
🔴	21.915	Previous Track 1
🔴	21.933	Next Playlist 4
🔴	22.413	Play Pause 2
🔴	22.585	Shuffle 6
🔴	22.736	Play Pause 1
🔴	22.749	Shuffle 10
🔴	23.095	Next Playlist 2
🔴	23.098	Shuffle 11
🔴	23.216	Shuffle 8

(b)

Figure 32: Examples of a match list for a (a) gesture with low (53%) goodness and (b) with high (100%) goodness. The low-goodness example has green dots for examples from other classes, and does not include all of its own class's examples (1–4); the high-goodness example includes all of its own and only of its own class examples with green dots.

class with every other example in every class. MAGIC displays this information on an example-by-example basis in the *match list* (Figure 28D). The match list displays a given example's distance to every other example, sorted by distance. The small dots to the left of each entry in the match list denote whether the distance to that example falls under the threshold for the given gesture or not. In Figure 28D, the threshold for *Wave* is set to 16.42, and the match list for example *Wave 4* is being shown. Every listing with a distance ≤ 16.42 is considered as a match to *Wave*, and shows a green dot; every other listing is not a match and has a red dot. Further examples are illustrated in Figure 32.

The *Recognized As* column in the interface (Figure 28C) uses the information from the match list to determine what class a given example would be recognized as. See the section on implementation (Section 5.5.3) for details on how the column is populated.

Along with the *Recognized As* column, MAGIC provides a summary of the match list by calculating a "goodness" value for each example. The details of how goodness is calculated are explained in Section 5.5.3; intuitively, an example has 100% goodness if all of the examples in its class are shown with green dots in the example's match list, and if *only* those from its class have green dots in the match list. A low goodness score may indicate a problem with an example, or with the class as a whole. MAGIC provides visualizations to assist the user in determining the source of low goodness scores.

It is important to note (as was emphasized to study participants) that goodness is not the main metric of success in gesture design. It is important that gestures be recognized reliably, but it is also important that false positives are not found when checking against the EGL. Gesture creation is a balancing act, which is one of the reasons it is so difficult.

One of the visualizations is of intra-class variability. Figure 31(a) is an example of a graph representing intra-class variability for the class *Wave*. Each numbered bar represents a single example in the class; in this case, six examples have been created. For each bar, the thick center line shows the average distance between that example and each other example in that class. A dotted line extends downward from the thick line to a circle at the bottom of the graph; this is simply so that the overall distribution of distances may be ascertained at a glance. The width of each bar represents the standard deviation of the distances to each other example in the class. The thicker dotted line with the number above it illustrates the recognition threshold for the class; the user may drag this line to interactively adjust the threshold. By looking at the graph, the designer is able to visually identify outliers. In this case, two examples—5 and 6—are quite different from the other examples, and might cause a low goodness score.

Figure 31(b)–(d) shows graphs visualizing inter-class variability for each class as compared to all other classes. In Figure 31(b), each shaded row represents a single class. The box in the row with a solid outline (on the left in each row) shows the mean and standard deviation of the intra-class distances, while the box with a dotted line (on the right in each row) shows the mean and standard deviation of distances between the class and all other classes. There is no relationship between the shaded rows of the table except that they are shown on the same numeric scale. This graph can be used to determine how confusable a class is with other classes. For example, class *Thump* shows very good differentiation from other classes, while class *Wave* is more confusable.

Figure 31(c) is similar to Figure 31(b), but splits the dotted-line box into its constituent classes. While the graph in 31(b) gives a general overview of how each class performs with respect to the other classes, this graph allows the user to determine if a single class is causing the problem. (With many classes, the graph can become very dense; Figure 31(d) shows a zoomed-in view of the row for class *Wave*.) The box that is the same color as the background represents the class in question; for *Wave* it is the bottom box. Each other box is color-coded according to the other classes, and shows the mean and standard deviation of the distances from (in this case) *Wave* to that class. A quick glance reveals that *Wave* and *Punch* (the top box in *Wave*'s row) are similar to each other. Looking at *Wave*'s box in the other rows of Figure 31(c) (the bottom box in every row), it can bee seen that the standard deviation of *Wave* with respect to each of these other classes is high.

5.4.4 Visualization: Gesture Testing

During gesture testing—both on the Testing tab and the EGL tab—the gestures created by the designer are compared with streams of pre-recorded data. In the Testing tab, the designer creates the streams; in the EGL tab, the sensor streams are recorded *a priori* by representative members of the designer's target user population. MAGIC provides visualizations of the results of the gesture search in two places.

The first is the *results list* (Figures 29D and 30D), which lists the matches between the testing stream and the gesture examples the designer has created. Each entry in the list gives the distance between the two examples, the time at which the match was found, and the name of the matching example.

The same information is visualized in the recorded sensor graph (Figures 29A and 30A). Each entry in the results list has a corresponding box superimposed on the sensor graph. The vertical space of the graph is divided into N slices, where N is the number of gesture classes defined; this allows overlapping boxes to remain distinguishable (this may be seen in Figure 30A). Clicking on a box in the recorded sensor graph highlights any matching results in the results list; the inverse is also true. Double-clicking in either location plays the video (Figures 33(b), 33(c), 30B) associated with that portion of the sensor stream.

Because it may often be the case that results from a comparison with the data in the EGL tab will number in the hundreds or thousands, it can be difficult to tell the root cause

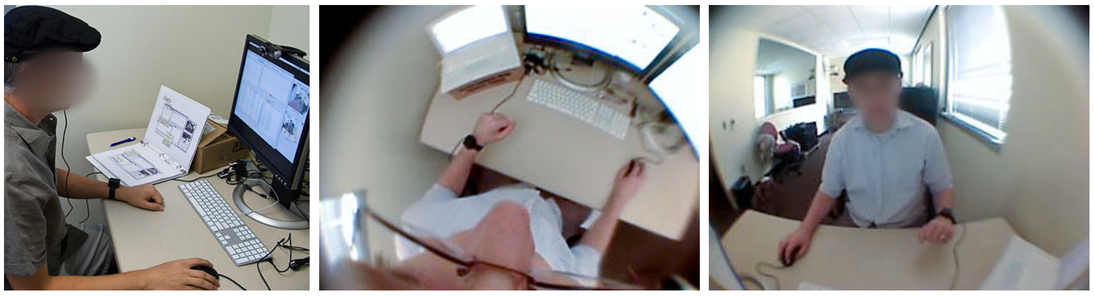

(a) (b) (c)

Figure 33: Experiment setup, showing participant wearing accelerometer and in-hat camera (a), view from in-hat camera (b) and view from on-monitor camera (c).

of the problem. MAGIC displays each match in the EGL as a black dot on the inter-class graphs, which is illustrated for the *Wave* class in Figure 31(b)–(d). The dots are plotted according to distance on the same horizontal scale as the rest of the graph elements, while the vertical scale is according to time. This display allows the designer to tell, at a glance: if there are many or few EGL matches; if all of the matches occur at one time or are evenly distributed throughout the EGL; and if the threshold for the gesture can be adjusted to remove most of the matches. (In Figure 31(d), the threshold cannot be moved to the left far enough to eliminate all EGL matches.)

5.5 Implementation Details

MAGIC is comprised of several interrelated pieces: the graphical user interface, the sensing backend, and the recognition backend. I used Python as the main language for MAGIC, due to its ease of use for prototyping. The GUI is implemented for the Apple Macintosh OS X operating system in Python Objective C, a Python-based bridge to the Objective C language in which OS X programs are written. Due to uneven threading support for Bluetooth-based applications in OS X, the sensing backend is a separate Python program that communicates with the GUI over local sockets.

5.5.1 Sensing and gesture recognition

MAGIC operates on time-series sensor data, and can accommodate a variety of sensors. For the purposes of this thesis, I used a wrist-mounted Bluetooth ±2G 3-axis accelerometer, sampling at 40Hz. Accelerometers are inexpensive and widely available, and are the current sensor of choice in consumer devices such as the Wii and iPhone. I chose to mount the accelerometer on the wrist, which I have shown to be a location that is easily accessible (Chapter 3) for control and display, and can allow for hands-free operation of devices.

When creating gestures, MAGIC automatically starts and stops recording movement. The starting and ending points of movement are detected by a simple threshold: if the average (over the three axes) variance of the last N points[1] exceeded a threshold value[2], recording started; when the same calculation yields less than the threshold, recording stops.

The start and stop points of the gestures are then refined using an exponential moving average (EMA) algorithm[3]; the same algorithm is used when initially loading the EGL to find "interesting" segments. For an acceleration sample at time t defined by $S_t =< x, y, z >$, the Euclidean distance is calculated as

$$D_t = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2 + (z_t - z_{t-1})^2}.$$

The exponential moving average E at time t is then calculated as

$$E_t = \alpha D_t + (1 - \alpha) E_{t-1},$$

where α is a smoothing factor; following Prekopcsák [77], I used $\alpha = .2$. Desirable data can then be taken at any t where $E_t > thresh$; I empirically determined $thresh = .2$ as the best value. Figure 34 illustrates the results of the EMA algorithm applied to a portion of the EGL.

MAGIC can utilize of a variety of gesture recognition algorithms. Due to its ease of implementation and relative computational efficiency (when optimized—see Section 5.5.2 for more details), I chose to use dynamic time warping (DTW). Given two signals, DTW returns a "distance", or difference, between them.

Figure 35(a) illustrates two continuous signals to be compared to each other using DTW. For example, the top (solid line) signal might be a recorded example from a 1D accelerometer, and the bottom (dotted line) signal might be an incoming sample that is to be compared.

[1] N was set to 10, or $0.25s$
[2] Experimentally calculated at .001
[3] `http://en.wikipedia.org/wiki/Moving_average`

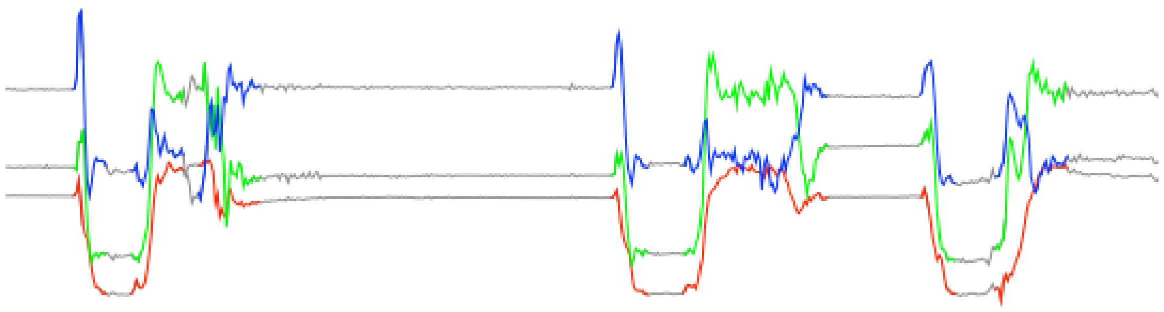


Figure 34: A segment of the Everyday Gesture Library, illustrating higher-energy segments—in color—that are checked for matches; the grey segments are skipped.

The signals look similar, but are not exactly the same; for example, the bump comes earlier in the lower signal. Figure 35(b) shows how DTW aligns the two signals. Essentially, DTW finds a way to transform the signals into each other with a minimum of stretching and shrinking. Each grey line connecting the two signals shows how DTW has determined that they correspond to each other. Notice how the grey lines illustrate that for the dotted line to become the solid line, the dotted line's bump must be shifted towards the the right. Figure 35(c) shows how a similarity score is computed. The signals are placed on top of each other in a way to minimize the lengths of the grey lines, and then the lengths of all of the grey lines are added up. Intuitively, if the two signals were exactly the same, the grey lines would all be of length zero. The further from zero the sum of the lengths is, the less good of a match between the two signals has been found.

I extended the basic DTW algorithm using the SWM (Scaled and Warped Matching) method of Fu *et al.* [24]. SWM scales a query sequence by factors from $1/s$ to s in order to provide better matching.

To perform recognition, MAGIC takes a potential gesture (a *candidate*) and, using DTW, compares it in turn to each recorded training example (each belonging to a particular class). MAGIC uses the three axes of acceleration as well as FFT-based features computed for each input sample. In order to be considered a match, an example's distance must first fall below a per-class threshold value. Each example falling under the threshold is considered, and the class with the overall lowest score according to weighted voting is considered to be the match. By default, the threshold is set automatically by MAGIC to

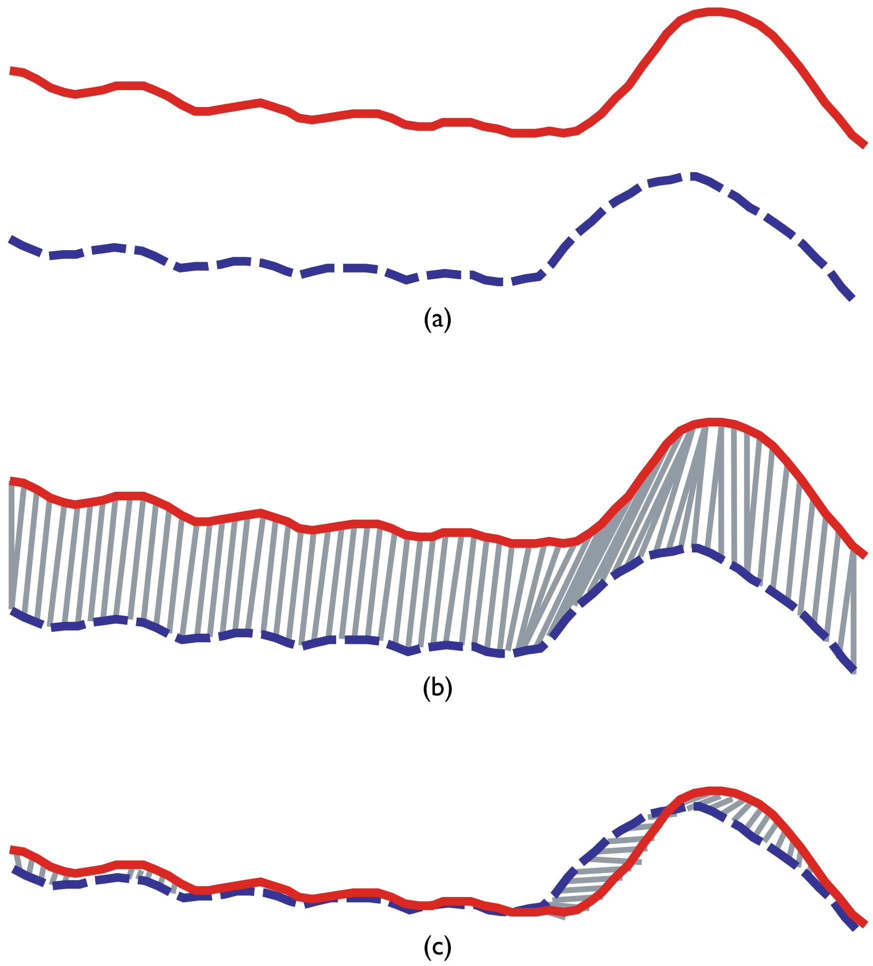


Figure 35: A graphical explanation of dynamic time warping (DTW). (a) shows two signals to be matched; (b) shows the best fit between the two signals, and by adding the length of the lines in (c) the similarity between the two signals is calculated. (Image adapted from [24]).

maximize the “goodness” value (discussed below) for the gesture class. If desired, the user may manually set the threshold, or may revert to the automatic behavior through use of an “Automatically Calculate Threshold” button.

In the EGL search process, if a training example is shorter than a segment of the EGL, MAGIC slides the training example along the candidate and finds the distance at each point. Matches are then computed in the same way as described above.

5.5.2 Optimization

I initially wrote the recognition backend to MAGIC in pure Python, but quickly discovered that it was extremely slow. Cython[4] allows C code to be mixed in with Python code, allowing code to be quickly prototyped and then incrementally "upgraded" to higher speeds.

I took several other steps to improve MAGIC's speed. The EGL search process was a major focus of these efforts. Because the EGL is searched using dynamic time warping, an $O(n^2)$ algorithm, it can be very slow. Compounding the speed issues is that, in order to elicit better performance, I implemented the SWM (Scaling With Matching) extension to EGL as described by Fu *et al.* [24].

One possible approach to speeding up DTW is using a lower bound and then performing an *early abandon* [64]; in the context of MAGIC, if a distance in the middle of a DTW computation ever exceeded the threshold for a gesture, the computation could be abandoned, since it could not possibly be a match. An issue with this approach, however, is that the threshold can change; it is adjusted both manually by users and automatically by MAGIC itself. Using the early abandon approach means re-searching large sections of the EGL if the threshold is ever adjusted to a lower value.

Therefore, I took a different approach to optimizing the EGL search. The first was adding a skip factor. When a gesture example is shorter than a piece of the EGL to be searched, MAGIC slides the example along the EGL segment and computes the distance at every point. Setting the skip factor to n allows MAGIC to jump n samples rather than sliding along every point within the EGL segment. I used a skip factor of 10, or 0.25 seconds; due to the warping nature of DTW, gestures are still detected when using this method. The second optimization step was to cache EGL results for each gesture example; then, whenever the threshold was adjusted, a simple $O(n)$ search was performed for each example to find locations where the EGL distance was lower than the new threshold. The only time a new EGL search must be done is when a new gesture example is added to the system.

[4]`http://www.cython.org`

Finally, the EGL search is a process extremely amenable to parallelization. Because the EGL is divided into discrete chunks by the EMA process (Section 5.5.1), and because each gesture example is independent of the others when it comes to searching, there are $C = N \cdot E$ separate computations to do, where N is the number of gesture examples to search for and E is the number of chunks the EGL has been divided into. These C computations can then be run in parallel; in my implementation, I used the *multiprocessing* Python library to launch 8 separate processes (one per processor core) and divided the C computations amongst the processes.

I also optimized distance computations between gesture examples by caching; in retrospect, I should have parallelized the search in the same way as the EGL, as multiple study participants complained about lag that was brought on by extensive example-to-example distance computation.

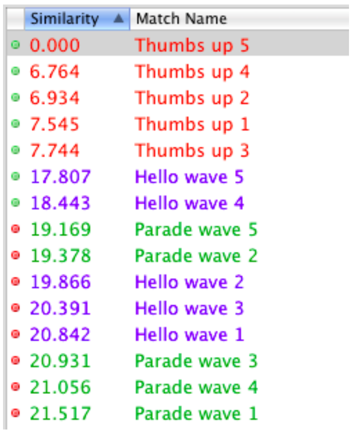
5.5.3 *Recognized As* and *Goodness*

The *Recognized As* and *Goodness* columns require some explanation as to their calculation.

The computation of *Recognized As* is best explained by an example. Consider the match list for the example *Thumbs up 5* as shown in Figure 36. Here, the threshold for *Thumbs up* (TU) has been set to 18.5, so there are seven green-dotted examples. Of these, five belong to *Thumbs up* and two to *Hello wave* (HW). To determine the class that *Thumbs up 5* belongs to, a score is calculated for each of the represented classes, the class with the minimum score is chosen. First, the similarity scores for the green-dotted examples are divided by class:

$$\begin{aligned} TU &= (0, 6.764, 6.934, 7.545, 7.744) \\ HW &= (17.807, 18.443) \end{aligned}$$

Next, the median value for each group is calculated:



	Similarity ▲	Match Name
●	0.000	Thumbs up 5
●	6.764	Thumbs up 4
●	6.934	Thumbs up 2
●	7.545	Thumbs up 1
●	7.744	Thumbs up 3
●	17.807	Hello wave 5
●	18.443	Hello wave 4
●	19.169	Parade wave 5
●	19.378	Parade wave 2
●	19.866	Hello wave 2
●	20.391	Hello wave 3
●	20.842	Hello wave 1
●	20.931	Parade wave 3
●	21.056	Parade wave 4
●	21.517	Parade wave 1

Figure 36: This Figure helps to illustrate how the *Recognized As* value is calculated for each gesture example.

$$\begin{aligned} TU_{med} &= 6.934 \\ HW_{med} &= 18.125 \end{aligned}$$

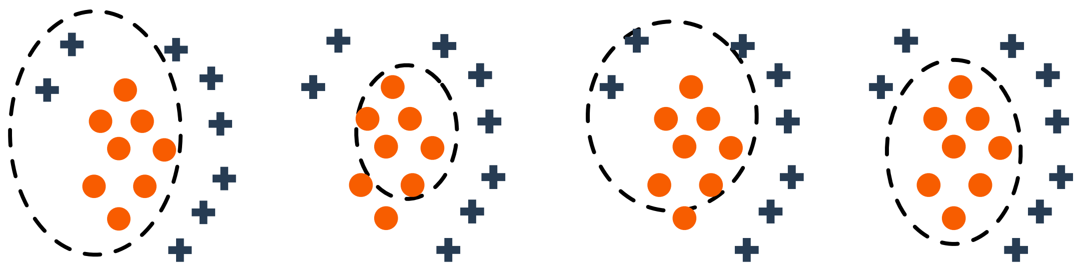
Then, the median value is divided by the proportion of the green dots that class makes up:

$$\begin{aligned} TU_{score} &= \frac{TU_{med}}{|TU|/(|TU|+|HW|)} = 9.708 \\ HW_{score} &= \frac{HW_{med}}{|HW|/(|TU|+|HW|)} = 25.375 \end{aligned}$$

Finally, the class with the lowest score is chosen as the *Recognized as* class; in this case, TU is the winner.

Note that other methods—such as k-nearest neighbor—could have been used here. The method I used was developed organically while writing the software, and was intended to cope with outliers while yielding intuitive results.

An example's *goodness* is based on the precision and recall for cross-validation (the example compared to every other example in all classes). In statistical machine learning, *precision* is the percentage of results labeled as a class that actually belong to the class and *recall* is the percentage of a class as a whole that was labeled as belonging to the class. The



(a) *p=.80, r=1.0, g=.89* (b) *p=1.0, r=.75, g=.86* (c) *p=.78, r=.88, g=.82* (d) *p=1.0, r=1.0, g=1.0*

Figure 37: These examples visually illustrate precision (p) and recall (g) and how they relate to goodness (g). The orange circles can be thought of as examples from a single class of gestures, and the blue crosses as examples from other classes. Any shape encircled by the dotted line is considered to belong to the class of orange circles. In (a), ten shapes are encircled, but only eight belong to the correct class, so the precision is 80%. However, all of the orange circles are included, so recall is 100%. The harmonic mean of these two numbers gives a goodness value of 89%. In (b), all of the shapes encircled are of the orange circle class, so precision is 100%; however, two of the eight members of the class were missed, and so recall is only 75%, giving a goodness of 86%. Illustration (c) illustrates both imperfect precision and recall, and (d) shows both at 100%.

F_1 measure, or goodness, is the harmonic mean of precision and recall:

$$\text{goodness} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Goodness ranges from 0–100%. Intuitively, an example only gets a goodness score of 100% if it matches *all and only all* of the other examples from its class. Figure 37 illustrates these ideas.

5.6 Conclusions

This chapter has presented the design and operation of MAGIC, a Multiple Action Gesture Interface Creation tool. The next chapter details the design and results of an evaluation of MAGIC.

CHAPTER VI

MAGIC: EVALUATION

In order to determine the efficacy of MAGIC, I conducted a user study. My goals in the study were to:

1. qualitatively assess the usability of MAGIC by observing users utilizing it to design gestures;
2. understand the design strategies used by designers when creating gestures;
3. determine how effective the Everyday Gesture Library is in helping users to design gesture sets that are unlikely to be accidentally activated by everyday movements; and
4. determine if different designers use the same movements for the same gesture commands.

6.1 Procedure

In order to understand the efficacy of the Everyday Gesture Library (goal 3), there were three experimental groups, illustrated in Figure 38. Each group received the same experimental task. The main difference was between group *noEGL* and groups *EGL* and *myEGL*: group *noEGL* received the MAGIC interface with the EGL tab removed, and participants in this group were not initially informed of the existence of the EGL. Groups *EGL* and *myEGL* both received the full interface.

Groups *noEGL* and *EGL* consisted of people with user-centered design experience, and group *EGL* used a standard EGL collected by the first author. Group *myEGL* was comprised of volunteers who collected EGL data (see Section 6.2), and each *myEGL* participant used their own EGL data during the study.

Each participant was seated at a desk in a chair without arms (to allow for free arm

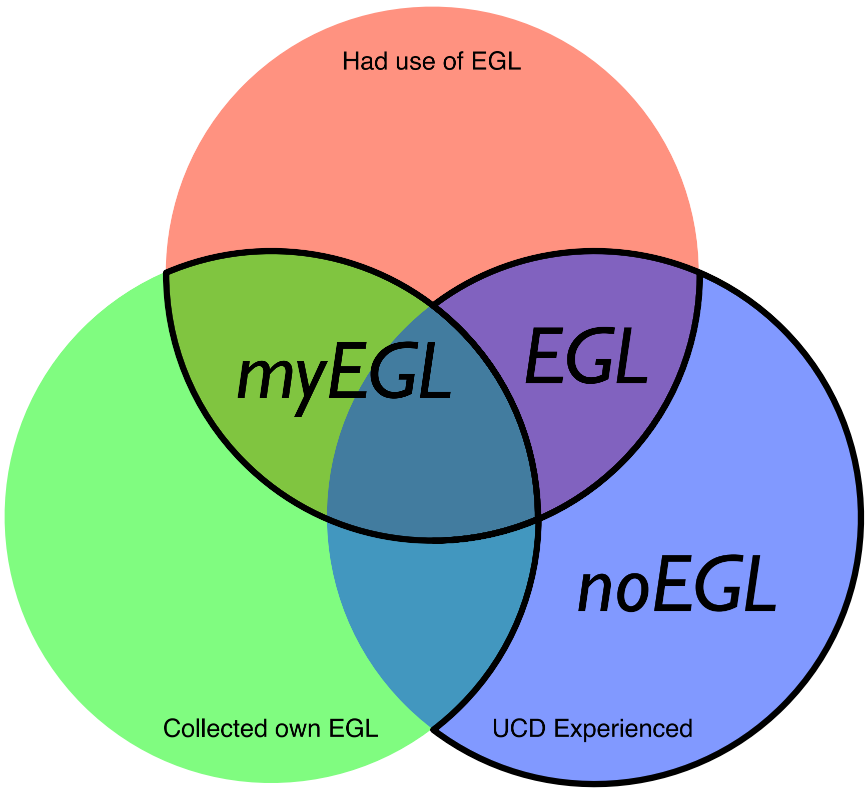


Figure 38: The three study conditions.

movement). To begin, the participant was requested to wear the hat with the camera and to wear the wireless accelerometer on the left wrist (Figure 33(a)).

Each participant worked through a tutorial (see Appendix E for the tutorial used for conditions *EGL* and *myEGL*; condition *noEGL*'s tutorial was identical aside from references to the EGL) walking them through use of the MAGIC software and giving a brief introduction to gesture recognition (similar to that presented in Section 5.5.1). The tutorial also included a section on troubleshooting, designed to assist users in solving common problems with gesture creation. This section was added after pilot tests revealed that users frequently got stuck with certain problems (for example, gestures with low goodness scores or with many EGL occurrences).

Participants took about an hour to complete the tutorial; afterwards, they were given the opportunity to ask any questions, and then were given a printout explaining the experimental task.

The printout asked participants to take on the role of an employee of the Gesture Engineering Group at "Pear" Computer (see Appendices C and D). Participants were requested to design and create eight gestures to control a new wrist-mounted, gesture-controlled digital audio player, the "uPod Touchless." The functions to control were:

- Play/Pause
- Next Track
- Previous Track
- Volume Up 10%
- Volume Down 10%
- Next Playlist
- Previous Playlist
- Shuffle

These functions were chosen because they are commands that would be plausible to control with gestures, and for a type of device that participants would be familiar with. Volume up and down were specified as "10%" in order to make a discrete command; MAGIC's gesture recognition does not currently work with continuous motions.

The instructions also asked participants to ensure that each gesture met the following criteria, chosen as requirements that would be important for creating an actual product:

- The gesture must reliably activate the desired function.
- Performing the gesture must not activate other functions.
- The functionality associated with a gesture must not be activated by a user's everyday movements.
- The gesture should be easy to remember.

- The gesture should be easy to perform.
- The gesture should be socially acceptable.

Participants were provided scrap paper and the tutorial, and were given approximately 2.5 hours to complete the task using the MAGIC software.

When participants in condition *noEGL* indicated that they were finished, the researcher compared the created gestures to the same EGL given to group *EGL*. The researcher then verbally informed the participant of the results, asking them to fix any gestures that had a large number of occurrences in the EGL. The goal was to simulate the current state of the art in gesture system design, in which a designer must equip one or more users with the device and allow them to use it for several days, reporting back with how it worked or did not.

At completion of the experiment, the researcher conducted a semi-structured interview with each participant, focusing on overall strategy of gesture creation and use of the software. After the interview, each participant was asked to complete a paper survey about the experiment (see Appendix B), comprised of the Questionnaire for User Interface Satisfaction (QUIS) [15], QUIS-inspired questions about the gestures, and a section requesting descriptions of why the participant chose the particular movements for each gesture.

6.2 EGL Creation

In order to create Everyday Gesture Libraries to test against, we recruited eight volunteers to collect everyday life data. Table 5 summarizes the volunteers and amount of data collected. Volunteers were compensated US$10 per hour of collected data. With the exception of myself (EGL1), the volunteers did not know any details about the study.

Each participant was provided with a data collection system and an instruction sheet detailing the use of the system. The data collection systems consisted of a wrist-mounted wireless accelerometer, an Asus eeePC 901 netbook computer, a shoulder-mounted bag or backpack, and a hat with a fisheye camera lens mounted in the brim, pointing downwards (see Figure 39 for representative images). Participants were requested to collect up to ten

Table 5: Summary data of EGL collection volunteers. Some volunteers collected more data than is represented here, but some data was discarded due to fragmentation and other hardware issues; only data actually used is shown.

EGL	Age	Gender	Occupation	Hours collected
EGL1	32	M	PhD CS	19:18
EGL2	26	M	MS CS	9:33
EGL3	27	F	Librarian	7:40
EGL4	28	M	Civil engineer	6:53
EGL5	30	F	IT professional	4:50
EGL6	37	M	IT professional	4:08
EGL7	28	F	Food writer	3:57
EGL8	29	F	Project manager	2:09

hours worth of data, and were asked to provide variation in the data so as to avoid ten hours of sitting at a desk or watching television.

Participants collected over 58 hours of data, encompassing activities such as attending a major academic conference, brewing beer, knitting, vacationing at the beach and in the mountains, working at the computer, hiking, cooking, performing home repair tasks, and attending work meetings.

Four of the volunteers (vEGL2, vEGL3, vEGL4, vEGL5) also volunteered for the MAGIC evaluation; due to hardware difficulties, the video data from one of the four (vEGL5) was lost and that participant used MAGIC without video associated with the EGL. These four volunteers comprise condition *myEGL*.

6.2.1 EGL Data Used in Evaluation

In order to provide EGL data for participants in condition *EGL* to use, and with which to provide feedback to condition *noEGL*, some data had to be taken from one of the collected EGLs. In pilot testing, I found that there appeared to be a high correlation between a smaller portion of an EGL and the rest of it, so during the evaluation I provided all participants with the same small portion of an EGL.

When creating classifiers, pattern recognition experts use a portion of data for training the classifier, and reserve the rest of the data for testing. Novices often forgo this step and end up *testing on training*, which gives invalid results: the idea is that the classifier has been tuned to the training data, so naturally will perform well on the same data. Providing

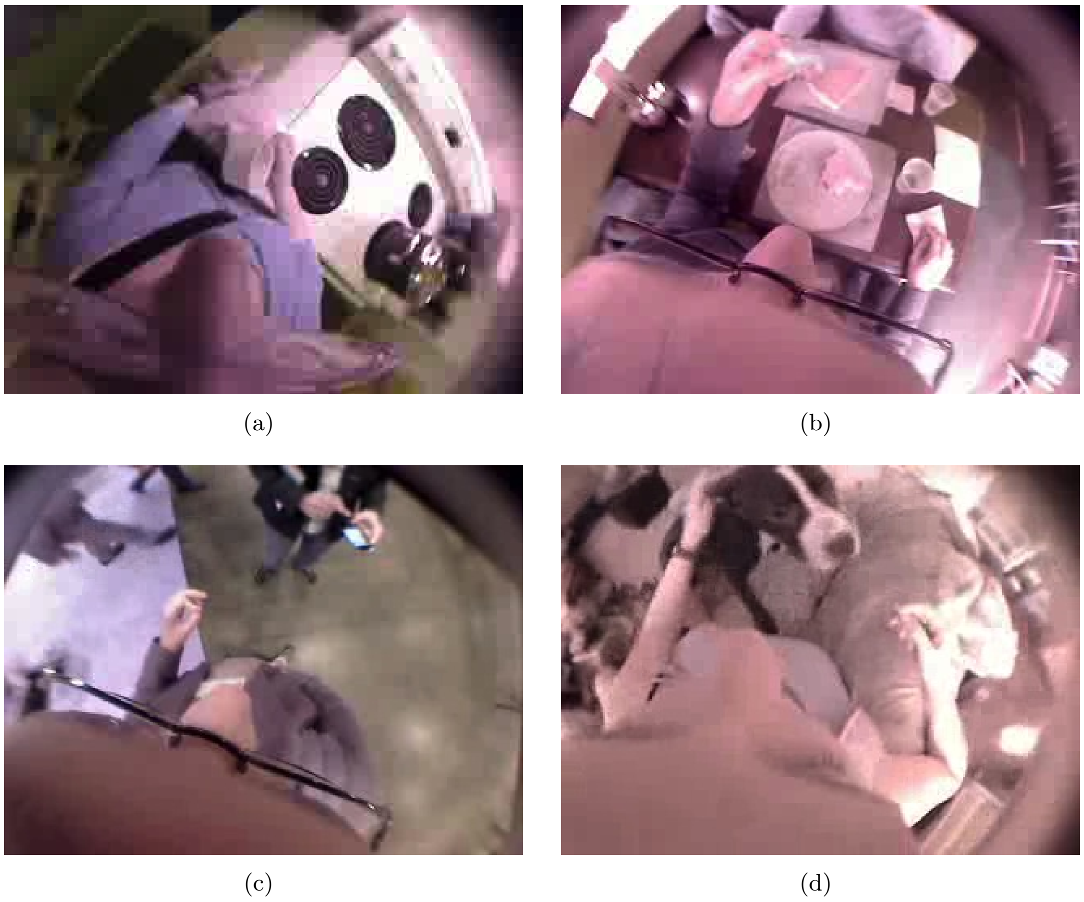


Figure 39: Sample images from EGLs: (a) cooking, (b) eating lunch, (c) talking with a colleague, (d) playing with a dog.

"fresh" data to the classifier will give results in line with what would be expected in the real world.

I used the same philosophy with the EGL. In the case of MAGIC, the user is the "classifier", and in the evaluation, I wanted to ensure that the results that participants tuned on would generalize to other situations. Therefore, I provided one segment of EGL1 ("EGL1 training") to condition *EGL* participants. The segment was simply one complete 5-hour collection. The remaining 14 hours of EGL1 ("EGL1 testing") were reserved for *post-hoc* testing (see Section 6.4.1.1). Participants in condition *myEGL* similarly received abbreviated training portions of their own EGLs.

6.2.2 EGL Limitations

An Everyday Gesture Library is subject to several limitations in terms of its generalizability. Essentially, an EGL *cannot* be guaranteed to span the entire space of everyday life, even for a single person. Too many new situations can arise that might not have been part of the EGL, even if an EGL was recorded over years. For example, a person might decide to go bungee jumping for the first time in their lives, and such a situation might cause a false positive. A less extreme example is moving between accommodations: a person might live in a house for ten years, and then decide to move. The motions made while packing and moving could be novel with respect to the EGL.

Aside from this limitation, an EGL is limited by three other factors; if any one changes, a new EGL is needed. The factors are *sensor*, *situation*, and *target population*.

The sensor refers to the actual sensor that is used—both for EGL collection and eventual product use—as well as the sensor's orientation and location on the body. If, for example, a gyroscope is to be used, an EGL collected with an accelerometer will not provide the correct data with which to test. Likewise, if an EGL has been collected with the sensor on the left wrist, moving to an application where the sensor is to be on the right wrist, or on the foot, will necessitate the collection of a new EGL.

Everyday Gesture Libraries should be collected according to the situation in which the end device is intended to be used. For example, an EGL based on several weeks' worth of recordings of high school teenagers would not be very useful for use in designing an in-cockpit gestural interface for fighter pilots; however, even the everyday activities of a fighter pilot might not be relevant. An EGL should be collected based on the use context of the planned interface; that is, it should include activities that might reasonably be expected to cause false positives in the actual usage situation.

Finally, the target population for the end product must be taken into account when creating an EGL. As with the fighter pilot example above, the planned users of the device may influence the situation. They may also influence the sensor or sensor placement; an EGL to help design interfaces for amputees to control prosthetic arms (such as DEKA's robotic arm[1]) may require gyroscopes for high precision, or may need for sensors to be placed on the foot for easy control. Finally, users with differing abilities will necessitate differing EGLs; teenagers will not be good models for elderly adults, and people with Parkinson's disease will provide very different databases than will someone with a broken arm.

6.3 Participants

I recruited a total of 16 participants; one participant took over two hours to complete the tutorial and was therefore discarded as an outlier, leaving 15 participants. My criteria for recruitment for conditions *noEGL* and *EGL* was a familiarity with user-centered design or building user interfaces; participants were mostly graduate students or recent graduates from the HCI and Computer Science programs at my institution. Condition *myEGL* participants were recruited based on a willingness to wear data collection hardware for extended lengths of time and included a student of library science, a civil engineer, an HCI graduate student, an IT professional, and a freelance food writer.

I requested participants to complete a brief demographic survey before starting the experiment (Appendix A). The questions included experience with motion-sensing devices such as the Nintendo Wii game system and the iPhone or T-Mobile G1 mobile phones. On a 9-point scale it included experience with user-centered design, with designing user interfaces, and with pattern recognition. See Table 6 for a summary of participant information.

[1]`http://www.dekaresearch.com/deka_arm.shtml`

Table 6: Demographic information for participants. Columns from left to right are: experimental conditions; number of participants; mean age; number of female participants; number of participants wearing a watch ("sometimes" responses counted as .5); experience with Nintendo Wii (1-9); experience with motion-sensing mobile phones (0-5); experience with user-centered design (1-9); experience in designing user interfaces (1-9); and experience with pattern recognition (1-9).

Cond	#	Age	#F	Watch	Wii	Phone	UCD	UI	PR
noEGL	7	29.0	2	3.5	3.2	2.0	5.9	6.7	4.0
EGL	8	31.6	2	2.0	3.4	1.4	6.6	5.6	3.0
myEGL	4	27.8	2	1.5	3.0	2.0	3.5	2.5	3.8

6.4 Results

In this section, I assume $p < .05$ for statistical significance. Participant 9 was determined to be an outlier due to EGL performance (greater than $2SD$ from the mean) and was not used in the statistics in this section.

6.4.1 User Performance

6.4.1.1 Everyday Gesture Library

There are two ways to think about—and therefore normalize—EGL data. One way is to normalize the number of occurrences in each EGL by the length of the EGL. This tell us how many false positives can be expected per length of time (i.e., per hour) for a normal day. These are the numbers reported in Table 8. The other way to normalize the EGL is by length of data actually considered. Because the EGL is segmented by "energy" (see Section 5.5.1), only a subset is actually searched. Normalizing by the length of data searched allows meaningful comparisons to be made between EGLs: therefore, I use this normalized number of hits for statistical computations.

As expected, the participants with access to the EGL (conditions *EGL* and *myEGL*) had significantly fewer EGL occurrences on the training EGL portions—an average of 1.9 per gesture per hour (SD = 9.6)—than those without (condition *noEGL*), who had an average of 52.1 (SD = 117.8) occurrences per gesture per hour. Five participants (one from condition *noEGL*, three from *EGL*, and one from *myEGL*) achieved zero occurrences in the EGL. (The *noEGL* participant performed multiple repeats of each of his gestures, such as making a circle three times.) Four (one *noEGL*, two *EGL* and one *myEGL*) had an average of one occurrence/gesture/hour, one (*EGL*) had four occurrences/gesture/hour, and the rest of the participants had more than five occurrences/gesture/hour. There was no statistically significant difference between conditions *EGL* and *myEGL*.

Clearly, 1.9 accidental activations per gesture per hour is not an acceptable number for a real system. However, looking at the gestures individually—rather than grouped by participants—nearly a third of the gestures created (50 of 152) had zero occurrences in any EGL (see Table 8)! Chapter 7.2.3 discusses how better sensing—leading to more reliable recognition—can improve on these results.

As mentioned earlier, five hours of EGL1 data (EGL1 testing) was used during the experiment. An independent-samples t-test on the number of occurrences in the reserved data revealed a significant difference between the two conditions. There was a significant 99.28% correlation between the number of occurrences in EGL1 training and EGL1 testing.

For the additional EGLs collected by other volunteers (EGL2–8), the difference in number of occurrences between the conditions *EGL* and *noEGL* was also found to be significant. The correlation between the number of EGL occurrences in the experimental portion of EGL1 and each of the additional EGLs was significant and greater than 87%.

The high rate of correlation between the part of EGL1 used for the *EGL* condition and the other EGLs implies that—at least for this particular combination of sensor, recognition algorithm, and task—that a small EGL is as useful as a large one.

The EGL proved a success in terms of avoiding gestures that might appear in everyday life, activating undesired functionality. In terms of this goal, participants were positive about the utility of the EGL and the usability of the interface. Some participants, however, experienced difficulty with finding gestures that did not occur in the EGL. In the words of one condition *EGL* participant, "I just kinda feared the EGL." Many participants, however, appeared to quickly learn from their mistakes and were able to create new gestures with few or no EGL occurrences in just one or two tries.

Surprisingly, few participants took advantage of the EGL video to determine what movements made by the subject of the EGL conflicted with the created gesture. In general, the reason stated by participants was that while the cause of the occurrences was intellectually interesting, it was not very useful in determining what to do next. As one participant commented, “I didn’t care *why* I was hitting the Everyday Gesture Library; I can’t change what’s in there!” Another reason for the lack of interest in the EGL video may be that watching that many video clips seemed overwhelming: even clicking to see the video for ten different occurrences would take a minute or two, and when thousands of occurrences were found, it makes sense for users to not even try.

Condition *myEGL* participants differed, showing interest in their recorded activities and how their created gesture examples matched; however, most said that the video did not help in understanding how to proceed in creating new gestures that would not conflict with the EGL.

Participant 24, in condition *myEGL*, had problems with her data collection system, which only recorded about two minutes of video. She therefore used MAGIC without EGL video, being able to see only the accelerometer recording. When asked in the post-experiment interview if she would have liked to see the video, she responded with a strong affirmative, saying that she thought it would have helped her to adjust her gestures so they would not conflict with the EGL.

6.4.1.2 Gesture Creation

Overall, participants were successful in the gesture creation task, with none failing to create distinct gestures for the eight “uPod” functions. On average, participants created 5.1 examples per gesture (SD=1.5); it is likely, however, that participants were influenced by the tutorial, which instructed them to make five examples for each gesture.

The mean goodness value for the gestures over all the participants was 86.3% ($SD =$ 23.5%), with three achieving 100% goodness for all eight gestures and seven with scores for all gestures above 97%. The poorest-performing participant had an average goodness (across all classes) of 68%, with the worst gesture having a 44.9% goodness. There was no significant correlation between the goodness of gestures and the experimental condition.

The implication of this high level of gesture goodness is that participants were able to use the software to design a set of eight gestures that had high internal consistency—that is, a movement intended to be of a particular gesture is likely to be recognized as such—and high differentiability—that is, a movement intended to be of a particular gesture is unlikely to be mistaken for a different gesture. Both consistency and differentiability are important factors for real-world gesture use.

Although participants were given no instructions as to how long to make their gestures, most opted for short motions. The average length of gestures was 2.6 seconds ($SD = 1.5$), which falls well under the four second threshold for microinteractions as discussed in Chapter 1.

6.4.1.3 Software Usage

On average, participants took two hours to complete the experimental task, regardless of the condition. This was somewhat surprising—prior to the study, I had surmised that *noEGL* participants would create a set of gestures in a fairly short amount of time and then be finished. However, participants in all conditions found it challenging to create a set of gestures with high goodness, and the *noEGL* group simply used the time to try to improve their goodness values (although they did not necessarily succeed; as related in Section 6.4.1.2 there was no correlation between experimental condition and goodness values).

Also contrary to my expectations, almost all participants proceeded through the task in a very linear manner, creating all of the classes and examples first, then trying them out on the Testing or EGL tabs, rather than creating one gesture at a time and testing each one. I expected the latter rather than the former because it would lead to less restarting of gesture creation. I frequently observed a participant discovering that a particular gesture had many EGL occurrences, and not only recreating the gesture but the related gesture, if any. For example, if Next Track had many EGL occurrences, and Previous Track was a mirror of the movement, the participant might re-create both gestures to maintain consistency, regardless of whether Previous Track had EGL occurrences or not.

Within the creation tab, the approach was also quite linear, with the usual flow being as follows: create a gesture class; record one or more examples for that class; troubleshoot the class if it has a low goodness; record any desired final examples; repeat by starting with a new class. Several participants followed an alternate approach of recording examples for all eight gestures initially, and then troubleshooting as a whole.

After observing almost all of the participants following the same pattern, I have come to the conclusion that the linear process may be a result of the layout of MAGIC. Until nearing the end of the process, participants tended to proceed in a tab-by-tab manner. A more unified interface might encourage users to consider different approaches. The tutorial also may be a cause; in order to provide clear explanation, it walked participants through the process in a linear fashion.

A second possible explanation for the linear approach is the responsiveness issues the system experienced as more gestures and examples were created. As shown in Section 6.4.2.1, participants rated MAGIC poorly with respect to system speed. The lack of responsiveness—especially when creating new gesture examples after a large number already exist—may have discouraged participants from iterating further than absolutely necessary.

Overall, participants used MAGIC in an exploratory and iterative fashion. Only one of the participants pre-planned gestures; the rest immediately started to experiment with different movements. As particular examples (or entire gesture classes) failed to conform to expectations, participants disabled or deleted the offending item and tried to create something that would work better. Participants moved fluidly between checking whether their gestures worked as expected on the testing and EGL tabs and modifying the gestures on the creation tab.

6.4.1.4 Performance Statistics

Statistically controlling for participant and experimental condition (using ANCOVA, the analysis of covariance), few predictors of performance could be found, either for goodness or number of EGL occurrences. Participants' self-rated pattern recognition experience (on a scale of 1–9) appeared to have a significant impact on the goodness of gestures. Using Spearman's rho non-parametric correlation coefficient to further test, the data show that participants with a self-rated expertise of 1 had significantly lower goodness scores than several (but not all) higher-rated groups.

Aside from this one statistic, there were no other predictors of EGL performance or goodness, either amongst the collected demographic data or the creation strategies used (Section 6.4.5). Further ANCOVA testing shows that gesture goodness is predicted by the person, and that number of EGL occurrences is predicted by the experimental condition.

6.4.2 Qualitative Results

Participant response to the software was very positive, with comments such as "gesture creation was easy" and "it's really fun." Some participants expressed some frustration with the difficulty of the task, both in terms of creating gestures with high goodness values and finding gestures that did not conflict with the EGL. One such participant commented, "I found the experiment pretty frustrating... [but] a relatively small fraction was due to the software itself."

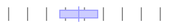
Although the tutorial included a section on troubleshooting—including advice on what to do when gestures have low goodness values or appear too many times in the EGL—few participants made use of it. When asked, several said that they had forgotten it was there. This suggests that such information would be better integrated into MAGIC itself, much like *quill*'s [54] automated advice.

6.4.2.1 Questionnaire Results

The Questionnaire for User Interaction Satisfaction (QUIS) is a 9-point Likert questionnaire in several categories. The QUIS was included in the questionnaire that participants completed after finishing the experiment. In addition to the standard set of QUIS categories, I added an *Experiment* section that asked a number of more specific questions about MAGIC. The results of the questionnaire are summarized in Table 7.

The mean (std) response to all questions was 6.9 (1.8), while the mean (std) response to only the questions in the original QUIS questionnaire (sans my added *Experiment* section) was 6.9 (1.7). The lowest mean response—and the only mean with a score less than five—was 4.7, to "System speed (too slow $\cdots$ fast enough)" in the "System Capabilities" category. The highest mean score, 8.4, was to "Use of terminology throughout the system (inconsistent $\cdots$ consistent)" in the "Terminology and System Information" category.

Table 7: The results of the questionnaire are summarized as *mean/std*, with the bar graph illustrating the same. The responses for each question are indicated in parentheses; the response on the left is scored as 1 and the response on the right as 9. Each section also had space for the participant to provide additional comments.

	mean / std	
Overall Reactions to the Software		
(terrible ··· wonderful)	7.3 / 0.9	
(frustrating ··· satisfying)	6.4 / 1.7	
(dull ··· stimulating)	7.2 / 2.0	
(difficult ··· easy)	5.8 / 1.4	
(inadequate power ··· adequate power)	7.0 / 1.5	
(rigid ··· flexible)	6.7 / 1.4	
Information Presentation		
Screen layouts were helpful (never ··· always)	7.5 / 1.1	
Amount of information that can be displayed on the screen (inadequate ··· adequate)	7.5 / 1.1	
Arrangement of information on the screen (illogical ··· logical)	7.6 / 1.1	
Sequence of screens (Gesture Creation/Gesture Testing/Everyday Gesture Library) (confusing ··· clear)	8.1 / 1.3	
Progression of work-related tasks (confusing ··· clearly marked)	7.1 / 1.8	
Terminology and System Information		
Use of terminology throughout system (inconsistent ··· consistent)	8.4 / 0.8	
Computer keeps you informed about what it is doing (never ··· always)	6.3 / 1.6	
Performing an operation leads to a predictable result (never ··· always)	6.8 / 1.3	
Information about quality of gestures (confusing ··· clear)	6.7 / 1.9	
Learning		
Learning to operate the system (difficult ··· easy)	6.9 / 1.8	
Time to learn to use the system (slow ··· fast)	6.4 / 2.0	
Exploration of features by trial and error (discouraging ··· encouraging)	7.0 / 1.8	
Tasks can be performed in a straight-forward manner (never ··· always)	7.1 / 1.6	
Steps to complete a task follow a logical sequence (never ··· always)	7.3 / 1.5	
System Capabilities		
System speed (too slow ··· fast enough)	4.7 / 2.0	
Correcting your mistakes (difficult ··· easy)	6.5 / 2.1	
Ease of operation depends on your level of experience (never ··· always)	6.9 / 1.4	
Experiment		
Match between designed gestures and intended purposes (low ··· high)	6.7 / 1.8	
Ease for someone else to learn designed gestures (low ··· high)	6.2 / 1.7	
I would like to own the uPod Touchless, to use with my designed gestures (no way ··· definitely)	6.7 / 2.8	

The *Experiment* section of the questionnaire was intended to discover whether the participants were satisfied with the gestures they had created and with the concept of gesture control as a whole. Participants were somewhat satisfied, with the median response being 7 to the first two questions in the section. Response to "I would like to own the uPod Touchless, to use with my designed gestures" saw some of the most divergent opinions; the median response was 8, but there were were four responses of three or less, and nine '9' responses. Interestingly, there was no correlation between the desire to own a uPod Touchless and the performance of the participant, either with the EGL or gesture goodness.

The Experiment section of the survey also had a free-response section which asked participants to consider if there were other devices they might like to control with gestures. Ten of the 19 participants responded, and the imagined devices were surprisingly varied. Several participants said they would like to control a mobile phone; other responses included day trading software, presentation control, stereo controls, devices for the visually impaired, and a "Roomba mess indicator".

The final section of the questionnaire asked participants to comment on their gestures: "We're interested in your design choices; for each gesture, please write a short description of why you chose the particular gesture for the given functionality. If you had a strong impression of what the gesture should be, but ended up not using that idea, please write down the idea and why it didn't work out. You can go back and watch your videos if you need a reminder of what the gestures looked like." Although many participants simply described their gestures, this section proved invaluable in conjunction with the recorded gesture videos for determining the strategies used, as discussed in Section 6.4.5.

6.4.3 Retrospection

Some of the aspects of MAGIC that the study revealed to be most important were those that enabled easy experimentation, both in the creation and testing of gestures. The key characteristic in this regard was *retrospection*—the ability to return to previously-created content and review the actions taken.

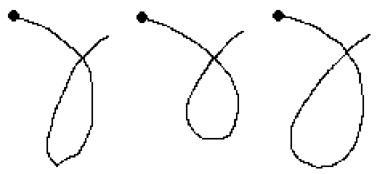


Figure 40: A visual comparison of three different pen gesture examples in Quill [54]; such visual comparisons are difficult for non-visual motion-based gesture systems. (Image redrawn from [54].)

This functionality is important when designing non-visual interactions, because the designer may forget the mapping between specific inputs and outputs; in my study, participants often forgot the motions they made to activate specific commands, especially during the initial period of trying out many different gestures.

During the post-experiment interviews, the feature that participants most frequently cited as being of primary usefulness was the video playback of gesture examples. This was due to the need for retrospection: by the time a participant reached the testing phase, or wanted to record some more examples for a particular gesture class (for instance, to improve its goodness score), she may have forgotten the movements that defined the gesture. In such situations, participants watched the video recordings of themselves making the gestures to remind themselves of the proper movements. Retrospection is also useful due to the nature of a non-visual recognition-based interface: in a visual system such as Quill [54], two pieces of training input may be visually compared to note their differences (Figure 40). In a motion-based system such as MAGIC, however, such visual comparisons are much more difficult.

In an example of what I term *retrospective realization*, some participants used the retrospective facilities in MAGIC to modify gestures based on unintentional movements. Several participants related occasions when one example of a class—thought to be nearly identical to the others—performed markedly better or worse in terms of the goodness score. In such cases, the participant watched the video for the affected example and two or three others of the same class to discern the difference. Frequently it was found that some unintentional movement had been included in the gesture; in some cases, where the example in question performed better than the others, the participant incorporated the unintentional movement into the gesture, re-recording the other examples to include the motion. Figure 41 illustrates retrospective realization for one participant.

The video saved along with each testing sample that was recorded was also found to be useful by many participants. While some performed gestures from only one class in each testing sample, other participants made multiple gestures in sequence. Frequently, some of the motions would fail to be recognized—either at all, or as from the correct gesture class—and the participant would watch the video to ascertain whether the movement had been made poorly or if it really should have been recognized. It was in such situations that many participants made use of the ability to select a portion of the test sample and add it as an example to a particular class, allowing free-form movements made during testing to be used *post hoc* as input to the system.

As can be seen in Figure 33, participants were recorded by both a hat-mounted and a monitor-mounted camera during gesture recording. Contrary to my expectations, the monitor camera was more popular amongst participants than was the hat camera. I anticipated that users would prefer a first-person view to help them understand and remember the movements they performed; however, many participants indicated they felt more able to understand their gestures from a forward perspective than from a top-down view.

The least-used visualization was the recorded accelerometer graph (Figure 28A). Similar to the graph of a sound wave in an audio program, participants were able to determine the magnitude of the movement but little else. The graph *was* of use to them, however: several participants mentioned using the appearance of the graphs to determine whether a newly recorded example sufficiently matched the previously recorded examples. Some users were able to glean more information by looking at the differently-colored axes—one participant mentioned being able to remember what a movement was like based on the directions implied by the colors—but most were unable to connect the shape of the three lines to the arm and wrist movements that produced them. It is interesting to note that, while this visualization was little-used, it is the visualization currently used in most systems involving interaction with time-series data [30, 101].

For participants in condition *myEGL*, an additional retrospection capability was available: the EGL recorded by each person. Condition *myEGL* participants reported that, because the EGL was personal, they took time to look at the portions of their EGL that their gestures conflicted with. Most *myEGL* participants said they liked the ability and would want to have it in the future, but admitted that it may not have been helpful to them in making gestures that did not occur in the EGL.

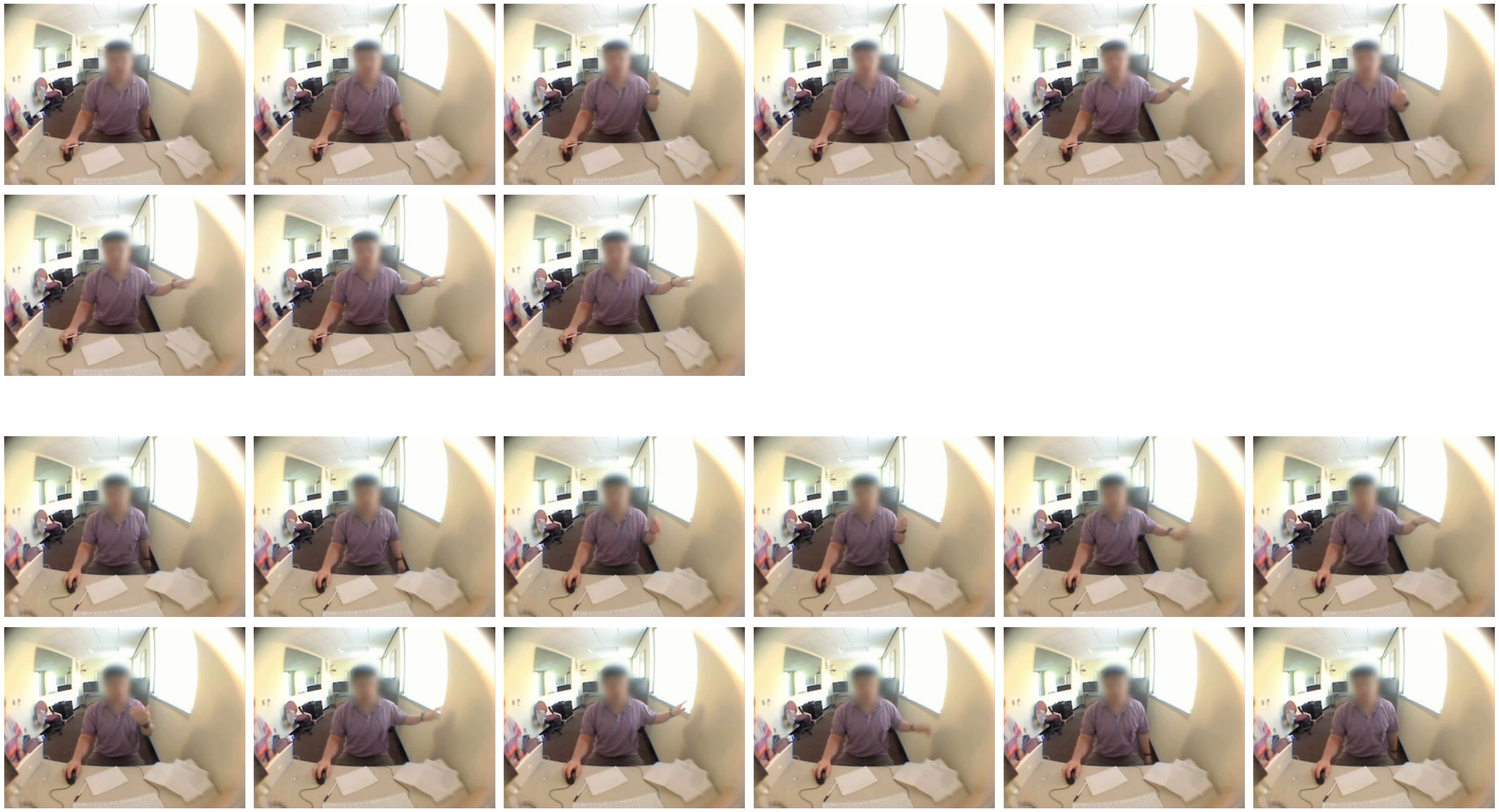


Figure 41: The two sequences of images show two examples of the *previous playlist* gesture for a participant, and illustrate *retrospective realization*. The top sequence shows the gesture as initially created—a double "sweeping" motion in the air. The bottom sequence shows the same gesture, but here the participant put his arm down before MAGIC stopped recording. When he discovered that the latter movement was much less confused with other gestures than the former, he re-recorded all of his *previous playlist* examples to include the arm-down portion.

6.4.4 Comprehension

In addition to aiding designers in understanding what *they* have done, MAGIC offers visualization features designed to help users understand what the *system* has done, including the visualizations and "Recognized As" and "Goodness" columns.. Overall, I consider the set of features a success. There was no feature that participants did not report using, likely due to the complexity of the task and the difficulty in understanding pattern recognition topics by non-experts in the domain.

Some participants found the live accelerometer graph (Figure 28B) useful, especially for experimenting with the effects of arm movements on the resulting shape of the trace. Contrary to my expectations, no participants reported looking at the live graph during actual example recording, instead remaining focused on the list of examples to see when the newly recorded example appeared.

Participants considered essential the statistics about the performance of individual gesture examples. These included the "Recognized As" and "Goodness" columns (Figure 28C), the match box showing similarity scores (Figure 28D), and the intra-gesture graph (Figure 31(a)). Several participants found that gesture examples that they intuitively felt to be exactly what they wanted were shown to be outliers by the goodness score or intra-gesture graph. A minority of users found the match box to be confusing, and few compared the numbers within the box to each other, instead relying on the ordering of examples.

The intra- and inter-class graphs (Figure 31(b)–(d)) were not as widely used. Many users found them confusing, or interpreted them in a way that was not consistent with the design. One of the major criticisms brought up was that these graphs were overwhelming, without the benefit of providing guidance on how to fix the problems revealed therein. Many users did state that they felt the graphs were "essential", however, implying that they should not be discarded, but tweaked for better usability.

6.4.5 Design Strategies

In an experiment asking participants to generate gestures for various commands using a tabletop display, Wobbrock *et al.* found that for many commands, participants generated the same or very similar gestures [107]. I did not find the same results for participants in the current experiment. Instead, gestures varied between participants, sometimes wildly.

In designing gestures, participants had several competing concerns, which became clear while observing participants over the course of the experiment. For those in condition *noEGL*, I observed the concerns to be (in decreasing order of importance to participants):

1. gestures should be easy to remember;

2. gesture examples should have high goodness scores;

3. gesture classes should have high average goodnesses;

4. testing samples should be recognized correctly; and

5. gestures should be socially acceptable.

Conditions *EGL* and *myEGL* added the EGL, and with it the further concern, most important of all to the participants, that

1. gesture examples should have few or no EGL occurrences;

with the other concerns shifted downward in priority accordingly.

These concerns influenced how participants interacted with the system and the strategies that were used to design gestures. Reviewing the videos recorded for each gesture example across all gesture classes and participants, I identified several ways in which the participants attempted to achieve these conflicting goals.

Some strategies were exercised solely for the sake of memorability, usually involving *iconic* gestures—movements intended to represent particular objects or visuals in the world. One of the most common iconics used by participant was making movements in the form of shapes—both simple and more complex—such as circles, letters, or familiar icons such as the play (▶) or pause (❚❚) symbols. More complex iconics were also used; a motion used by several participants for the Next and Previous Playlist commands was to mime turning pages in a book.

Another strategy frequently used for memorability was *pairing*—defining gestures for two related commands in such a way that the motions are also related. An example is the

Next Track and Previous Track commands: a gesture for Next Track might be to wave the hand to the left, while the Previous Track gesture might be waving the hand in the opposite direction.

Frequently, motions used for memorability were modified or composed with other motions in order to influence recognition by improving goodness scores or testing performance, or to reduce the number of occurrences found in the EGL. A very common strategy was to *repeat* a motion multiple times to make it more distinguishable, for example by tracing a circle in the air twice. Another approach was to add an *impact* somewhere in the movement, such as hitting one hand with another, or by snapping the fingers. Motions were also modified by adding another movement component, for example by adding abrupt stops (*jerks*) or *directional changes* to the motion. A special case of this approach is the common "shake to Shuffle" motion that several participants implemented.

Some participants developed the idea of a trigger motion. In the same manner that one might unlock a mobile phone by pushing a particular sequence of buttons, the participants designed either *pre-* or *post-fix* motions that were common across all gestures, indicating that the motion was intended to be a command to the system. For example, one participant prefixed every command with an ear-cupping motion, while another ended every motion with a confirmatory wrist-twist.

A few strategies appeared to be intended solely to prevent occurrences in the EGL. These tended to involve extremes, either in motion or in time. One participant quickly developed a strategy of shooting his arm outstretched at a sharp angle, and then drawing symbols in the air (Figure 42). Another participant made very deliberate motions with several repetitions of the motion (Figure 43).

The most successful gestures—in terms of low numbers of occurrences in the EGL—fell into two categories. Many were of the "extreme" type just described. Other successful gestures, however, involved impacts or near-impacts in the form of direction changes. One participant clapped his hands once, rotated the gesturing hand, and clapped again with the back of his hand. The same participant also had success in short, sharp movements, with abrupt directional changes (Figure 44).

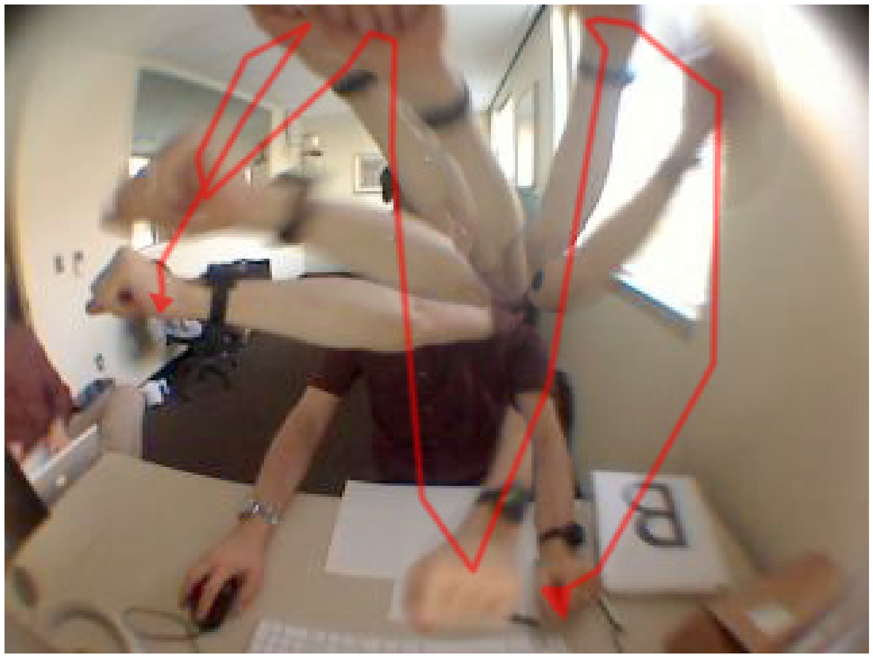


Figure 42: Participant 5 performing the Volume Up gesture, which consists of the arm moving straight up, tracing a “V” for “Volume” and then tracing a “+” for “increase”.

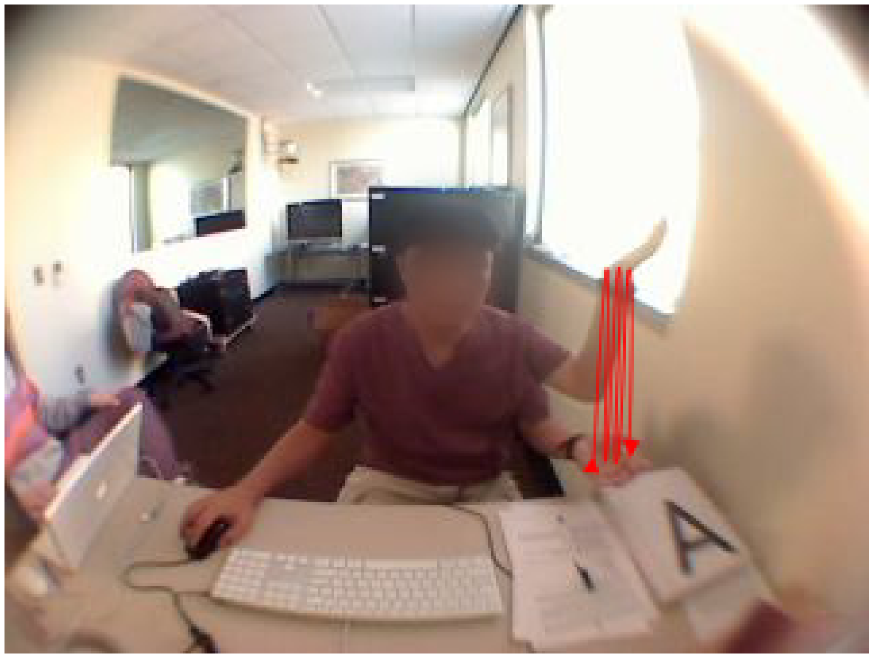


Figure 43: Participant 10 performing the Volume Up gesture, which consists of raising and lowering the hand three times, with the palm turned up.

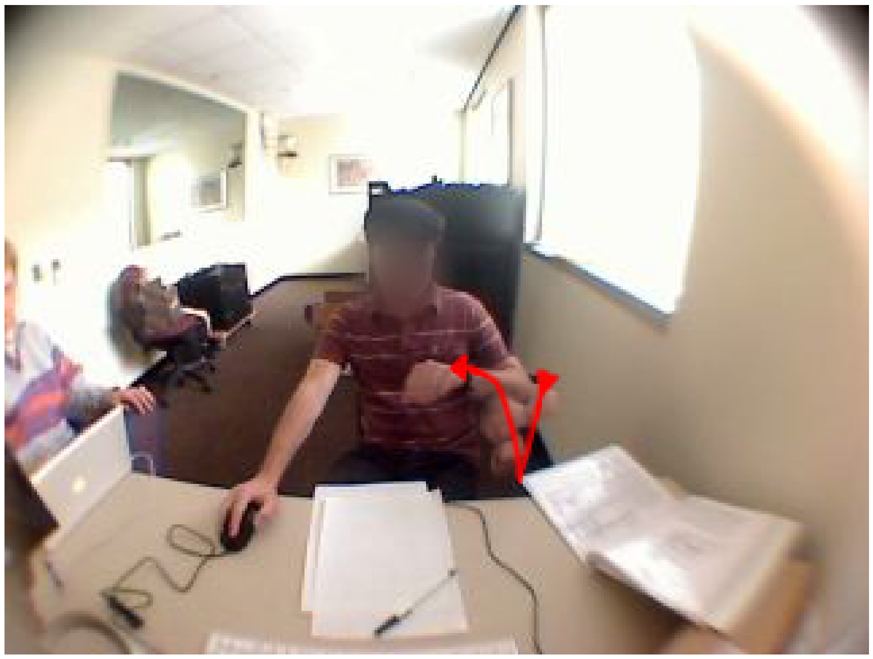


Figure 44: Participant 7 performing the Next Track gesture, which consists of quickly moving the hand down and then across the body, with a sharp directional change.

The various strategies used by participants can be broken down into six categories of movements; many gestures are composed of multiple motions, or have a single motion that falls into multiple categories (for example, drawing the play triangle ▶ is both iconic and involves sharp directional changes). I determined the categories by watching all of the videos recorded when participants created gestures, and as such the categorization is subject to observer error. The categories are as follows:

Iconic movements are those that make reference to some concept or symbol. An example is drawing a triangle in the air to represent the play ▶ symbol.

Directional movements are those that have meaning to the direction: many of the paired next/previous gestures included left/right components.

Prefix/postfix movements precede or follow other movements without being meaningful for the gesture itself. In practice, this mean that a participant pre- or postfixed every gesture with a common motion.

Impacts occur when the gesturing hand encounters a solid object such as the table, the

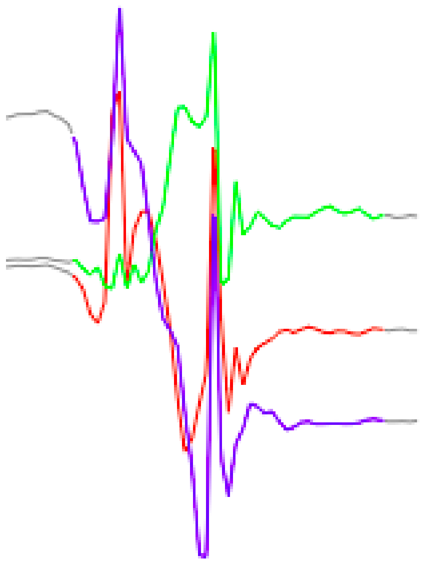


Figure 45: The acceleration graph of participant 7's Previous Playlist gestures, which consists of two quick claps, with the left hand on top for the first clap and on bottom for the second.

gesturer's body, or the gesturing hand itself (with a snap or a flicking motion). Impacts make a distinct spike in acceleration that may be clearly seen on the acceleration graph (Figure 45).

Repeated motions were common, especially as strategy to avoid occurrences in the EGL. Examples include drawing the same iconic multiple times or making several directional movements (Figure 43).

Jerks and directional changes produce acceleration spikes similar to those elicited by impacts. Jerks start or end a gesture with an abrupt motion, while directional changes are sharp deviations from the current course.

Table 8 lists the movement types for each gesture by each participant. Although many participants show similar patterns, almost no two gestures are alike. The one commonality was using a shaking motion for the Shuffle gesture; participants 9, 16, 18, 19, 22 and 24 all used a variation on this movement.

The numbers presented in Table 8 bear some explanation with respect to how the gestures have been standardized. Some participants were confused and made separate "Play" and “Pause” gestures, in which cases the gesture with the lower number of EGL occurrences was chosen to represent “Play/Pause”. In other cases, participants made extra gestures: one created a garbage class to hold false positive portions of testing gestures; another (condition *noEGL*) participant created his own “EGL” gestures. For these participants, the standardization process affects the goodness values shown in the table. Additionally, a bug in MAGIC influenced the number of EGL occurrences: some gestures shown in the table with many occurrences for EGL1 actually had zero occurrences for the participant during testing (“TestEGL” column).

Table 8: Gestures categorized by movement types. P# indicates participant number and Cnd condition (or EGL in the case of condition *myEGL*); movement types are **Ic**onic, **D**irectional, starting or ending with a **P**refix/**P**ostfix, having an **Im**pact, with **R**epeated movements, and containing a **J**erk or directional change. **Len** is the average length of each gesture example in the class in seconds, on a 0–10s scale. The thick red line indicates four seconds, under which a gesture qualifies as a microinteraction. **Goodness** shows the average goodness of a gesture class, from 0–100%. The "# EGL hits" columns show the number of occurrences in each of the indicated EGLs, per hour (normalized by total length of data; see Section 6.4.1.1 for more information). For participants in EGL conditions, the number of occurrences per hour on the EGL used in the interface is shown under "TestEGL" (this is EGL1 Test for *EGL* participants). The number in this column for participant 21 was lost due to a disk error.

		Strategies								# EGL occurrences								
P#Cnd	Gesture	Ic	D	PP	Im	R	J	Len (s)	Goodness (%)	TestEGL	EGL1	EGL2	EGL3	EGL6/8	EGL4	EGL7	EGL5	Total
5/*EGL*	Play/Pause	●		●			●	(2.5)	(85)	0	1	0	0	0	0	0	0	1
5/*EGL*	Next Track	●		●			●	(3.2)	(38)	0	0	0	0	0	0	0	1	1
5/*EGL*	Previous Track	●		●			●	(2.8)	(100)	0	0	0	0	0	0	0	0	0
5/*EGL*	Next Playlist	●		●			●	(2.6)	(68)	0	0	0	0	0	0	0	0	0
5/*EGL*	Previous Playlist	●		●			●	(2.1)	(97)	0	0	0	0	0	0	0	0	0
5/*EGL*	Volume Up	●		●			●	(3.1)	(72)	0	0	0	0	0	0	0	0	0
5/*EGL*	Volume Down	●		●			●	(2.3)	(87)	0	0	0	0	0	0	0	0	0
5/*EGL*	Shuffle	●		●			●	(2.6)	(88)	0	115	25	23	30	25	17	17	252
6/*EGL*	Play/Pause		●			●		(2.7)	(96)	0	0	0	0	2	1	0	0	3
6/*EGL*	Next Track		●			●	●	(2.2)	(100)	0	0	0	0	0	0	0	0	0
6/*EGL*	Previous Track		●			●	●	(2.4)	(100)	1	10	1	2	5	3	1	3	25
6/*EGL*	Next Playlist		●			●		(3.1)	(69)	0	2	1	0	3	0	1	2	9
6/*EGL*	Previous Playlist		●			●		(3.3)	(100)	0	0	0	0	0	0	0	0	0
6/*EGL*	Volume Up		●			●	●	(2.8)	(81)	0	0	0	0	0	0	0	0	0
6/*EGL*	Volume Down		●			●	●	(2.2)	(100)	0	1	0	0	0	0	1	0	2
6/*EGL*	Shuffle		●			●		(2.7)	(74)	0	0	0	0	0	0	0	0	0

		Strategies								# EGL occurrences								
P#Cnd	Gesture	Ic	D	PP	Im	R	J	Len (*s*)	Goodness (%)	TestEGL	EGL1	EGL2	EGL3	EGL6/8	EGL4	EGL7	EGL5	Total
7/*EGL*	Play/Pause						●	(1.5)	(100)	0	4	3	1	6	2	0	1	17
7/*EGL*	Next Track		●				●	(1.4)	(100)	0	0	0	0	0	0	0	0	0
7/*EGL*	Previous Track		●				●	(1.5)	(100)	0	0	0	0	0	0	0	0	0
7/*EGL*	Next Playlist		●		●			(1.3)	(100)	0	0	0	0	0	0	0	0	0
7/*EGL*	Previous Playlist		●		●			(1.4)	(100)	0	0	0	0	0	0	0	0	0
7/*EGL*	Volume Up				●		●	(1.6)	(77)	0	0	0	0	0	0	0	3	3
7/*EGL*	Volume Down				●		●	(1.5)	(100)	0	0	0	0	0	0	0	0	0
7/*EGL*	Shuffle					●	●	(1.8)	(100)	0	0	0	0	0	0	0	0	0
8/*EGL*	Play/Pause				●		●	(1.3)	(100)	0	1	0	0	1	0	0	0	2
8/*EGL*	Next Track							(2.2)	(95)	22	129	39	39	70	36	24	30	367
8/*EGL*	Previous Track	●					●	(1.2)	(100)	1	6	5	1	7	2	1	1	23
8/*EGL*	Next Playlist	●	●			●		(7.4)	(100)	0	0	0	0	0	0	0	0	0
8/*EGL*	Previous Playlist	●	●			●		(6.9)	(100)	0	0	0	0	0	0	0	0	0
8/*EGL*	Volume Up	●	●			●		(7.1)	(100)	1	3	1	0	1	0	1	1	7
8/*EGL*	Volume Down	●	●			●		(4.9)	(100)	10	42	4	9	8	4	15	13	95
8/*EGL*	Shuffle	●				●		(6.5)	(100)	0	0	0	0	0	0	0	0	0
9/*EGL*	Play/Pause	●					●	(1.9)	(100)	0	2	0	0	0	0	0	0	2
9/*EGL*	Next Track	●	●				●	(1.8)	(98)	6	236	45	32	38	28	73	18	470
9/*EGL*	Previous Track	●	●				●	(1.8)	(75)	54	506	93	95	131	86	78	59	1048
9/*EGL*	Next Playlist		●				●	(1.9)	(91)	156	1256	230	253	713	318	192	176	3138
9/*EGL*	Previous Playlist		●				●	(1.9)	(90)	3	473	140	170	213	142	108	77	1323
9/*EGL*	Volume Up	●	●			●		(2.2)	(82)	32	636	125	260	118	102	188	196	1625
9/*EGL*	Volume Down	●	●			●		(2.2)	(98)	0	1	1	0	0	0	0	0	2
9/*EGL*	Shuffle	●					●	(2.0)	(100)	70	441	183	141	297	154	84	189	1489
10/*noEGL*	Play/Pause	●				●	●	(4.1)	(79)	0	89	9	9	53	24	4	7	195
10/*noEGL*	Next Track		●			●		(3.9)	(100)	0	0	0	0	0	0	0	0	0
10/*noEGL*	Previous Track		●			●		(3.8)	(100)	0	0	0	0	0	0	0	0	0
10/*noEGL*	Next Playlist		●			●		(4.7)	(100)	0	0	0	0	0	0	0	0	0
10/*noEGL*	Previous Playlist		●			●		(4.4)	(100)	0	0	0	0	0	0	0	0	0
10/*noEGL*	Volume Up		●			●	●	(4.8)	(80)	0	16	1	3	21	3	12	7	63
10/*noEGL*	Volume Down		●			●		(4.4)	(84)	0	0	0	0	0	0	0	0	0
10/*noEGL*	Shuffle					●		(4.5)	(95)	0	0	0	0	0	0	0	0	0

		Strategies								# EGL occurrences								
P#Cnd	Gesture	Ic	D	PP	Im	R	J	Len (*s*)	Goodness (%)	TestEGL	EGL1	EGL2	EGL3	EGL6/8	EGL4	EGL7	EGL5	Total
12/*noEGL*	Play/Pause						•	(1.6)	(67)	0	3	3	3	7	3	3	3	25
12/*noEGL*	Next Track		•		•			(1.1)	(50)	6	270	32	51	129	36	29	34	581
12/*noEGL*	Previous Track		•		•			(1.1)	(52)	28	421	109	110	62	42	68	40	852
12/*noEGL*	Next Playlist		•					(1.5)	(63)	0	113	36	80	120	21	68	91	529
12/*noEGL*	Previous Playlist		•					(1.4)	(74)	0	51	6	8	46	4	25	16	156
12/*noEGL*	Volume Up	•	•				•	(2.3)	(66)	22	1352	495	576	703	609	321	391	4447
12/*noEGL*	Volume Down	•	•				•	(2.0)	(49)	19	2768	613	714	1443	817	457	476	7288
12/*noEGL*	Shuffle					•	•	(3.0)	(100)	0	0	0	0	0	0	0	0	0
13/*noEGL*	Play/Pause	•					•	(1.9)	(69)	7	15	4	4	14	4	3	4	48
13/*noEGL*	Next Track	•	•					(1.6)	(100)	32	1158	184	201	706	271	154	166	2840
13/*noEGL*	Previous Track	•	•					(1.4)	(71)	3	1800	592	1569	545	387	610	743	6246
13/*noEGL*	Next Playlist	•	•					(1.6)	(85)	20	98	47	224	70	44	48	33	564
13/*noEGL*	Previous Playlist	•	•					(1.5)	(100)	0	1	1	0	1	1	0	0	4
13/*noEGL*	Volume Up	•	•					(1.7)	(77)	11	44	15	54	18	5	7	17	160
13/*noEGL*	Volume Down	•	•					(1.8)	(100)	6	11	1	3	3	0	5	4	27
13/*noEGL*	Shuffle	•				•		(3.3)	(100)	1	1	1	1	1	1	0	0	5
14/*noEGL*	Play/Pause				•	•		(3.3)	(100)	0	2	0	1	0	1	1	1	6
14/*noEGL*	Next Track		•			•		(3.2)	(43)	0	1750	392	531	432	332	625	563	4625
14/*noEGL*	Previous Track		•			•		(3.1)	(66)	0	9	1	3	0	1	1	1	16
14/*noEGL*	Next Playlist		•			•	•	(2.8)	(95)	1	0	0	0	0	0	0	0	0
14/*noEGL*	Previous Playlist		•			•	•	(4.2)	(61)	0	400	176	735	195	258	217	298	2279
14/*noEGL*	Volume Up		•			•		(3.5)	(100)	0	0	0	0	0	0	0	0	0
14/*noEGL*	Volume Down		•			•		(3.4)	(29)	0	5383	1811	3807	2101	1966	2151	2551	19770
14/*noEGL*	Shuffle					•	•	(2.7)	(100)	0	0	0	0	0	0	0	0	0
15/*noEGL*	Play/Pause	•				•	•	(2.6)	(95)	0	2	1	4	0	1	0	0	8
15/*noEGL*	Next Track		•					(1.8)	(82)	25	121	48	156	43	27	34	19	448
15/*noEGL*	Previous Track		•		•			(1.1)	(100)	0	15	7	5	14	2	8	4	55
15/*noEGL*	Next Playlist		•				•	(1.5)	(95)	18	188	105	232	49	47	58	52	731
15/*noEGL*	Previous Playlist					•	•	(2.0)	(100)	0	0	0	0	0	0	0	0	0
15/*noEGL*	Volume Up		•				•	(1.4)	(100)	0	0	0	0	0	0	0	0	0
15/*noEGL*	Volume Down		•				•	(1.4)	(100)	31	609	79	76	624	116	50	117	1671
15/*noEGL*	Shuffle					•	•	(1.8)	(100)	1	0	0	0	0	1	0	0	1

P#Cnd	Gesture	Strategies						Len (*s*)	Goodness (%)	# EGL occurrences								
		Ic	D	PP	Im	R	J			TestEGL	EGL1	EGL2	EGL3	EGL6/8	EGL4	EGL7	EGL5	Total
16/*noEGL*	Play/Pause				●		●	(2.0)	(68)	32	57	14	65	15	4	11	53	219
16/*noEGL*	Next Track		●					(1.4)	(69)	69	288	113	985	73	49	64	217	1789
16/*noEGL*	Previous Track		●					(1.1)	(74)	3	77	21	70	20	5	19	43	255
16/*noEGL*	Next Playlist	●	●					(1.6)	(84)	1	0	0	0	0	0	2	0	2
16/*noEGL*	Previous Playlist	●	●					(1.2)	(74)	0	0	0	0	0	0	0	0	0
16/*noEGL*	Volume Up		●					(1.2)	(86)	3	27	9	66	9	3	5	35	154
16/*noEGL*	Volume Down		●					(1.6)	(64)	32	277	89	398	117	56	55	141	1133
16/*noEGL*	Shuffle	●				●	●	(3.6)	(98)	7	27	12	35	7	2	3	63	149
17/*EGL*	Play/Pause					●	●	(5.1)	(100)	0	0	0	0	0	0	0	0	0
17/*EGL*	Next Track							(3.9)	(100)	0	0	0	0	0	0	0	0	0
17/*EGL*	Previous Track		●			●		(3.9)	(100)	0	2	0	1	0	0	0	0	3
17/*EGL*	Next Playlist	●					●	(2.5)	(76)	0	0	0	0	4	0	0	0	4
17/*EGL*	Previous Playlist	●					●	(1.8)	(100)	0	1	0	0	3	0	0	0	4
17/*EGL*	Volume Up	●					●	(2.8)	(95)	0	0	0	0	0	0	0	0	0
17/*EGL*	Volume Down	●					●	(1.8)	(100)	6	44	9	3	27	13	4	7	107
17/*EGL*	Shuffle	●						(2.7)	(64)	1	122	36	22	61	47	27	22	337
18/EGL3	Play/Pause						●	(2.4)	(100)	47	217	57	136	111	98	104	46	769
18/EGL3	Next Track		●					(3.3)	(100)	0	0	0	0	0	0	0	0	0
18/EGL3	Previous Track		●			●		(2.8)	(95)	4	5	2	2	4	8	5	2	28
18/EGL3	Next Playlist		●					(3.0)	(100)	54	328	48	38	223	69	52	55	813
18/EGL3	Previous Playlist		●					(3.3)	(65)	1366	24	14	2	18	10	5	5	78
18/EGL3	Volume Up		●					(2.8)	(100)	0	0	0	0	0	0	0	0	0
18/EGL3	Volume Down		●		●			(2.3)	(100)	0	0	0	0	0	0	0	0	0
18/EGL3	Shuffle					●	●	(3.1)	(95)	66	336	71	74	60	45	49	44	679
19/*EGL*	Play/Pause	●		●		●	●	(3.2)	(66)	0	0	0	0	0	0	0	0	0
19/*EGL*	Next Track	●	●	●		●	●	(3.9)	(33)	0	0	0	0	0	0	0	0	0
19/*EGL*	Previous Track		●	●		●	●	(4.5)	(100)	0	0	0	0	0	0	0	0	0
19/*EGL*	Next Playlist	●	●	●		●	●	(4.0)	(40)	0	0	0	0	0	0	0	0	0
19/*EGL*	Previous Playlist		●	●		●	●	(3.1)	(40)	0	0	0	0	0	0	0	0	0
19/*EGL*	Volume Up		●	●		●		(4.0)	(100)	0	0	0	0	0	0	0	0	0
19/*EGL*	Volume Down		●	●		●		(4.2)	(100)	0	0	0	0	0	0	0	0	0
19/*EGL*	Shuffle	●		●		●	●	(4.8)	(69)	0	25	7	10	5	8	9	8	72

P#Cnd	Gesture	Ic	D	PP	Im	R	J	Len (*s*)	Goodness (%)	TestEGL	EGL1	EGL2	EGL3	EGL6/8	EGL4	EGL7	EGL5	Total
20/*noEGL*	Play/Pause	●					●	(1.7)	(61)	5	1571	821	1203	729	641	375	665	6005
20/*noEGL*	Next Track		●				●	(1.4)	(100)	17	118	139	7	17	42	7	36	366
20/*noEGL*	Previous Track		●				●	(1.2)	(100)	1	36	23	0	63	23	4	8	157
20/*noEGL*	Next Playlist	●	●					(2.4)	(100)	0	0	0	0	0	0	0	0	0
20/*noEGL*	Previous Playlist	●	●					(2.2)	(100)	0	3	1	4	2	9	2	2	23
20/*noEGL*	Volume Up		●					(1.8)	(62)	21	473	192	991	153	127	123	194	2253
20/*noEGL*	Volume Down		●					(1.9)	(58)	80	1254	428	850	478	352	254	311	3927
20/*noEGL*	Shuffle	●				●	●	(2.8)	(62)	45	831	251	1764	314	353	349	531	4393
21/*EGL*	Play/Pause						●	(6.1)	(53)		23	7	18	15	14	27	12	116
21/*EGL*	Next Track						●	(5.1)	(84)		1869	557	1042	779	652	721	693	6313
21/*EGL*	Previous Track						●	(2.4)	(97)		18	3	5	15	9	4	4	58
21/*EGL*	Next Playlist					●	●	(8.0)	(100)		0	0	0	0	0	0	0	0
21/*EGL*	Previous Playlist					●	●	(3.0)	(81)		0	0	0	0	0	0	0	0
21/*EGL*	Volume Up		●			●	●	(3.6)	(87)		0	0	0	0	0	0	0	0
21/*EGL*	Volume Down		●			●	●	(3.4)	(75)		2	0	0	1	1	1	0	5
21/*EGL*	Shuffle						●	(5.2)	(100)		0	0	0	0	0	0	0	0
22/EGL4	Play/Pause			●			●	(1.6)	(100)	1	3	1	3	1	4	1	0	13
22/EGL4	Next Track		●	●			●	(1.7)	(100)	0	1	0	1	0	0	0	0	2
22/EGL4	Previous Track		●				●	(1.6)	(100)	3	2	0	0	4	0	1	0	7
22/EGL4	Next Playlist		●	●		●	●	(2.6)	(100)	0	61	11	15	10	5	13	27	142
22/EGL4	Previous Playlist		●			●	●	(2.8)	(100)	0	0	0	0	0	0	0	0	0
22/EGL4	Volume Up	●				●		(3.0)	(100)	0	0	0	0	1	0	0	0	1
22/EGL4	Volume Down	●				●		(2.9)	(76)	0	72	18	422	14	6	6	18	556
22/EGL4	Shuffle					●	●	(2.0)	(100)	0	0	0	2	0	0	0	0	2
23/EGL2	Play/Pause	●					●	(1.2)	(97)	0	0	0	0	0	0	0	0	0
23/EGL2	Next Track		●				●	(1.6)	(95)	0	2	0	0	1	1	0	0	4
23/EGL2	Previous Track		●				●	(1.5)	(100)	0	0	0	0	0	0	2	0	2
23/EGL2	Next Playlist	●					●	(2.0)	(100)	0	2	1	0	0	1	1	1	6
23/EGL2	Previous Playlist	●					●	(2.1)	(100)	0	0	0	0	0	0	0	0	0
23/EGL2	Volume Up		●					(1.2)	(93)	0	0	0	0	0	0	0	0	0
23/EGL2	Volume Down		●					(1.2)	(90)	0	2	2	0	1	0	0	0	5
23/EGL2	Shuffle						●	(1.7)	(100)	0	0	0	0	0	0	0	0	0

		Strategies								# EGL occurrences								
P#Cnd	Gesture	Ic	D	PP	Im	R	J	Len (s)	Goodness (%)	TestEGL	EGL1	EGL2	EGL3	EGL6/8	EGL4	EGL7	EGL5	Total
24/EGL5	Play/Pause				●			(1.5)	(95)	1	103	17	8	14	10	29	35	216
24/EGL5	Next Track		●					(1.2)	(84)	17	240	28	10	232	17	17	39	583
24/EGL5	Previous Track		●					(1.2)	(72)	5	75	7	4	4	4	18	7	119
24/EGL5	Next Playlist		●			●	●	(1.6)	(83)	4	131	4	5	8	8	10	3	169
24/EGL5	Previous Playlist		●			●	●	(1.5)	(76)	3	73	9	8	3	5	13	7	118
24/EGL5	Volume Up	●	●			●		(1.6)	(81)	12	219	67	41	42	36	43	29	477
24/EGL5	Volume Down	●	●			●		(1.4)	(69)	12	76	14	19	5	4	18	16	152
24/EGL5	Shuffle					●	●	(2.0)	(51)	1	5	1	2	3	1	5	1	18

6.5 Discussion

Overall, I consider MAGIC to be a success. It received positive comments from the participants in the study, and users were able to complete the experimental task with high rates of success. MAGIC's support of retrospection was highly popular, and although many participants had some difficulty with the graphs, the visualizations were usable for comprehension.

In Chapter 5.2, I introduced a number of desiderata for a motion-gesture design tool. I now revisit these items and discuss how well MAGIC fulfills them.

allow non-expert use. I consider MAGIC to be a success on this point. As can be seen in Table 6, the participants were not expert in pattern recognition; however, all were able to successfully use MAGIC to design gestures, and those in condition *EGL* were often successful in creating gestures that did not conflict with the Everyday Gesture Library.

allow expert use. Some facilities exist in MAGIC for a user more expert in pattern recognition to adjust system behavior: users can pick between examples to use, and can adjust the recognition threshold for any gesture. The three participants who rated themselves as expert or near-expert in pattern recognition also appeared to have an easier time understanding the intra/extra graphs; this makes sense, as those graphs present familiar information such as inter- and intra-class variance.

encourage iteration. Participants used MAGIC in an iterative manner. Most participants heavily iterated on their gestures during the creation phase, especially when trying to gain a high goodness score. Participants also iterated on single gesture classes when working with the EGL, trying to change a gesture so that it had fewer occurrences. Most participants did not, however, change many or all of their gestures at once. This behavior may be due to the time and effort involved in gesture creation and troubleshooting; given a longer period of time, I would expect users to go through many different gesture sets.

support retrospection. MAGIC's support for retrospection was heavily used, and widely regarded by participants as essential. Some users requested more support for *comparative retrospection*; they wanted to watch two more more videos simultaneously, or view multiple recorded sensor graphs overlaid. Retrospection was also used to discover how unintentional movements incorporated into a gesture example influenced the example's goodness and EGL performance.

support further testing. Both the EGL and recorded video support *post-hoc* testing. The EGL allows a designer to test new gestures on a wide variety of subjects, in dozens of different circumstances. The video that is recorded alongside each gesture example is not only useful for retrospection during design, but could also be used for training others to use the gestures, or to test for others' reactions in terms of social acceptability.

6.6 Conclusion

In this chapter, I described the evaluation of MAGIC. I found that MAGIC was usable by my participants, that they enjoyed it, and that they were able to accomplish the experimental task. The EGL support in MAGIC helped users to create gestures with lower false positive rates than could those without the EGL.

In the next chapter, I will discuss the various limitations in MAGIC revealed by the experiment, and what future research avenues could be taken.

CHAPTER VII

MAGIC: IMPLICATIONS FOR DESIGN

The evaluation of MAGIC imparted many lessons, both specific to the implementation of MAGIC, and generalizable to other sensor-based system design tools. In this chapter, I discuss the lessons learned from the design, implementation and evaluation of MAGIC, changes and enhancements I intend to make in future work, and how the lessons from MAGIC can be generalized to other software.

7.1 Future Work

There are a number of features that would enhance the usability of MAGIC, but that I did not have time to implement.

The ability for more complex annotation was a frequently requested feature. Many participants made use of the gesture name to include notes or mnemonics about the gesture, such as "Pause (||)" or "next playlist (flip forward 90)". Multiple users requested further capabilities, such as making notes for each gesture example or testing sample, or annotating the testing tab accelerometer graph with the correct gesture for comparison with the system's results.

Although not addressed directly by participants, I realized during observation of their actions that the three-tab design should be reconsidered. Several times I watched a participant rapidly switch back and forth between the creation tab and the testing or EGL tab, making a small adjustment to a gesture and then observing its results. It also may be the case that the three-tab design locked participants into a particular workflow, encouraging them to first create all of the gestures and examples, then test, and then use the EGL. While SUEDE featured a similar three-part interface, modeled around a design/test/analyze paradigm [41], MAGIC is primarily a design tool; as such, users may benefit from a unified interface. Caution in design is called for, however; while Quill featured a single-stage interface, Long contemplated that "it may be useful to adopt a more workflow-oriented approach, such as Klemmer and colleagues used in SUEDE" [54].

As many of the participants remarked, gesture creation is a surprisingly challenging task, especially for non-experts. Several expressed a desire for more guidelines in creating gestures, hoping for guidance as to what sort of motions would make gestures with high goodness and low rates of EGL occurrences. As far as I am aware, no such guidelines exist. However, given the data I now possess, it may be possible to create an automated gesture advisor, not unlike Long's Quill system for pen gestures [54].

Lacking specific guidelines on motions, I *can* provide automated behavior in other areas. For example, very few participants manually adjusted the threshold—preferring the default automatic behavior—and most of those who did quickly reverted to the "Automatically Calculate Threshold" button. The threshold calculation routine simply optimized for the highest overall goodness score for a given class, but there is no reason that it could not further take into account the EGL and attempt to minimize the number of matches as well.

The tutorial (Appendix E) included troubleshooting tips for gesture creation ("My gesture has low goodness", "My gesture example is misrecognized"), testing ("The wrong things are recognized in my testing sample") and the EGL ("There are too many EGL hits"). At the least, this information should be included as part of the program. Many of the conditions under which troubleshooting is needed (low goodness, misrecognized example, EGL hits) can be automatically detected, and relevant help information could be provided.

Many participants requested further abilities to compare gesture examples to each other. Such a facility could be generalized to compare gesture examples against testing samples and the Everyday Gesture Library as well. Gestures are represented with acceleration traces and video. Using dynamic time warping, the best match between each sample of two acceleration traces may be determined. This information could then be used to warp the graphical representation of the gestures to show how the computer interprets them; it could also be used to play two videos simultaneously, aligning them according to the best time warp. Implementing these visualizations could help users understand how two gestures match (or fail to match) each other, and perhaps help them to understand how to modify their movements to achieve a better result.

The Testing tab was confusing to some participants, and essential to others. One improvement would be to take inspiration from Exemplar [30] and make the testing graph live, such that it always shows recognition results. This change might help users better understand the testing portion of MAGIC and create more robust gestures.

7.2 Shortcomings

There are a number of shortcomings in MAGIC that I plan to address in a future version. These can be roughly divided into the categories of performance, visual, and mechanical. I also discuss the shortcomings of the experiment.

7.2.1 Performance Shortcomings

In the manner of all alpha software, MAGIC was occasionally crash-prone. Early on I took the precaution of having it automatically save all of the gesture and testing data created by the participants any time a change was made, which allowed me to quickly recover from crashes, restore the state of the program, and allow the participant to proceed with the experiment.

More seriously, MAGIC suffered from severe slowdowns when many gesture examples were created. Despite caching, combinatorial calculations which were necessary to update the goodness value for a gesture example when added: each new example requires an $O(n^2)$ DTW distance check against every other example. This check often slowed the system to a crawl for tens of seconds. Such slowdowns were the source of the majority of complaints I received about the system in the post-task interview and on the questionnaire; user dissatisfaction was reflected in the low (4.7) QUIS score for "System speed (too slow $\cdots$ fast enough)" recorded in Section 6.4.2.1. Unlike the EGL search, the inter-example distance calculations were not parallelized; clearly they should be.

The EGL search is also a very processor-intensive task. Most EGL searches took between 5 and 25 seconds to complete, depending upon the length of the query gestures. The EGL search was a user-directed task, where the user had to explicitly click a "Check" button to initiate it. I chose this mechanism due to an assumption that users would create a small number of gesture examples for one gesture and then proceed to check the EGL.

The evaluation proved this assumption wrong; in fact, most users concentrate the creative gesture creation task at the beginning, and only later check the gestures against the EGL. This behavior suggests that automatically searching the EGL in the background, as gestures are created, is a better model that will lead to a more responsive interface.

7.2.2 Visual Shortcomings

The difficulty experienced by some participants in understanding the intra- and inter- graphs (Figure 31) points to the need for more research into the best way to visualize the information presented therein. Many participants gave up entirely on those graphs, instead gleaning the desired information from the match box or from the goodness scores; however, the information is not entirely redundant, and I believe there is value in communicating it.

The match box (Figure 28D) itself was found to be overwhelming by some. I did not observe any user scrolling in the box to see higher-distance matches; in fact, more than one user commented that they simply looked at the first few lines in the match box to see whether the first few examples were correctly classified. Despite some difficulty, however, a majority of participants felt the match box to be essential to the task.

Maintaining consistency in gesture coloring was a concern of some participants. In particular, they were reluctant to delete gesture classes (when replacing a non-working motion with a new one), preferring to delete each example within the class in order to keep the same color for the command.

7.2.3 Mechanical Shortcomings

I consider mechanical shortcomings to be those which involve the operation of MAGIC itself, and influence the completion of the task. Several of these considerations center around the gesture recognition itself.

One issue that was frustrating to participants was encountering too many occurrences in the Everyday Gesture Library. Frequently matches were non-intuitive. For example, the candidate gesture might be a swift movement of the hand across the body, traversing 2–3 feet of space; this motion might match a very subtle movement in the EGL, such as the movement of the hand from the keyboard to the trackpad on a laptop. Clearly, these two motions should be considered to be different.

One solution to this problem is to improve the recognition. Although dynamic time warping is no longer the method of choice for gesture recognition—often being replaced by techniques such as hidden Markov models [101]—it is still being successfully used [30, 42, 51, 98, 108], and techniques from current implementations could be incorporated to improve MAGIC's performance.

Another method to improve performance is to add additional sensors. The problem with accelerometers, as used in the evaluation, is that low-amplitude movements look very similar to each other—at least from the perspective of recognition algorithms. As discussed in Chapter 5.5.1, dynamic time warping tries to find the best way to match two signals, and so a gentle movement to the left may match a gentle movement to the right more closely than a violent leftward movement.

Non-acceleration-based sensors, offer the possibility of better recognition performance. Gyroscopes are now appearing in many devices and, in conjunction with accelerometers, can provide more precise sensing capabilities. For example, the Dutch company Xsens[1] sells a complete wireless motion-capture system based on accelerometers and gyroscopes. Microphones also present interesting possibilities for sensing [101].

7.2.4 Study Shortcomings

While I feel that the experiment accomplished my goals to assess MAGIC's usability, understand users' design strategies, and determine the efficacy of the EGL, my evaluation was lacking in some areas due to its preliminary nature. A primary area to be improved upon is the amount of time given to the software: 3.5 hours was a long time for participants to spend at once, but not as much time as I wanted for them to achieve expert status with the system. Further studies should be divided over multiple days to understand how users interact with the software longer term, especially as they become more expert in its use.

Another issue with my study is the lack of variability in gesturing situations. All of my participants were seated during gesture creation and testing, while the EGL was recorded

[1] `http://xsens.com`

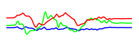
in many different mobility situations. I plan to make a mobile variant of the data collection system to allow designers to move about as they record gestures.

7.3 Implications for Design

Many of the lessons learned during the evaluation of MAGIC can be generalized to other interactive, sensor-based interface design tools. In this section, I discuss some of those lessons and how they may be applied.

In the course of the evaluation, participants revealed many strategies for designing memorable gestures. As discussed in other gesture research, gestures that are iconic are highly memorable [62]; therefore, it is not surprising that 14 of 19 participants used an iconic motion for at least one gesture. Related to iconicity is directionality, which was another popular strategy; similarly, many participants paired related gestures (for example, Volume Up involving a raising motion and Volume Down a lowering motion). Although no strategy was shown to lower the number of occurrences in the EGL, no strategy raised occurrences either; therefore, it may be feasible to use the strategies employed by participants as general suggestions for designers in the future.

One of the most important features to study participants was retrospection, which helped them to understand and compare their previous input. The direct manipulation principles of having a *continuous representation of the object of interest*, and of *naïve realism* [32] are relevant here. In a pen-based system, the strokes on the screen are the object of interest and are continually visible to the user; likewise, they "feel" like actual ink to the user—the strokes look and behave (at least at a basic level) like actual strokes from an actual pen. In such a system, there is no need for explicit retrospection functionality: the output rendering on the screen looks exactly like the input.

In a non-visual input system, however, naïve realism is lost, and with it continuous representation. Although time-series signals such as acceleration, sound, or gyroscopic angle can be represented in a graphical form, such an output form does not resemble the input, creating a gulf of evaluation. As such, either the user must train herself to understand "" as equivalent to her input, or the system must bridge this gulf and provide alternate output to assist in her understanding. In MAGIC, that alternate output attempted to restore the sense of naïve realism by providing video; in this way, the output again resembled the input. Because video is represented over time, it cannot provide a truly *continuous* representation of the motion made by the user (however, see Romero *et al.* [82] for visualizations of the movement of people over time); the users of MAGIC were able to overcome this lack, however, and effectively use the video for retrospection.

Another lesson from MAGIC relates to representing complex information. As discussed in Section 5.4.3, MAGIC re-presents the standard confusion matrix in multiple ways: the "recognized as" and "goodness" columns for each gesture example (Figure 28C); the match box (Figure 28D); and the intra-class and inter-class variability graphs (Figures 31(a)–(c)). Some participants used all of these sources of information, while most used only a subset. By providing a layering of simple-to-complex data, MAGIC allowed users to work with the information that they were most comfortable interpreting. While a single experiment is not enough to show this, I believe that such a gradient of information complexity may encourage a smooth novice-to-expert transition.

In a complex task such as the one that the participants in the evaluation were asked to do, there is a lot of information to be absorbed and simultaneously considered. More than one participant described the task as "overwhelming." This difficulty can help explain why so many participants found the "Automatically Calculate Threshold" button useful: it allowed them to worry about one less thing. Several participants mentioned that, for all of the information MAGIC gave them, it only showed them what was wrong with their gestures; it did not help them understand what to change to make better ones. These two findings suggest that automation can be a useful part of sensor-based design interfaces. Although [65] cautions that automatic interfaces can be frustrating to the user when unpredictable, giving users an explicit "automatically perform task" button can allow the user to maintain the feeling of control.

Offering automated advice may be useful in such situations as well. Although the notorious Microsoft Office Assistant "Clippy" has made many suspicious of automated help tools, following the same user-driven approach as I suggested above for automatic calculations could remove the distracting factors associated with such tools. A simple "help me with this gesture" button could trigger a help system giving suggestions for the most likely causes of problems.

CHAPTER VIII

CONCLUSIONS

As I stated in Chapter 1, my thesis statement is:

1. *Wrist-mounted sensors can be used to create gesture-based interface where the interactions occur in under four seconds.*

2. *Providing interaction designers with databases including peoples' everyday motions allows them to create gestures with fewer false positives than can be created without the databases.*

I showed part 1 in Chapters 4, 5 and 6. Chapter 4 introduced the idea of using the finger to interact with a round touchscreen wristwatch. In that chapter, I showed how to determine the optimal button layout for a given error rate, and demonstrated that users can activate functions in under four seconds. In Chapters 5 and 6, I introduced a Multiple Action Gesture Interface Creation tool, MAGIC, and showed that it could help designers create motion gestures that take a short amount of time to activate. Despite not being given instructions to keep the gestures short, the majority of users in my study of MAGIC created gestures well under four seconds long.

Chapter 6 also showed part 2 of my thesis. The Everyday Gesture Library functionality of MAGIC was successful in assisting designers in creating gestures with low rates of false positives. Several users were able to tune their gestures to have no false positives on the provided EGL, and very low rates on the other collected EGLs.

In addition to the two parts of my thesis statement, I made a number of other contributions centering around the concept of microinteractions and suggesting approaches of how they may be achieved. First, I defined microinteractions as *interactions with a device that take less than four seconds to initiate and complete.* The microinteraction concept is a useful lens through which to view mobile interaction, and can help steer mobile HCI research in a direction that may yield devices more usable in difficult situations such as being on-the-go. Through the Quickdraw study, I demonstrated that that placing devices on the wrist allows them to be quickly accessed, even while walking. These results provided justification for investigating the wrist as a platform for interaction.

I explored microinteractions on the wrist using a touchscreen watch. I investigated ways to interact with a round touchscreen watch with a finger, rather than a stylus, and introduced the idea of rim-based interaction for the watch. My exploratory study showed how to calculate error rates and button sizes for three different types of movement, and I illustrated interaction concepts.

Finally, I investigated motion gesture as a method for performing microinteractions, and introduced MAGIC—a tool for designing low false-positive gestures. I presented an evaluation of the software, and discussed future directions. MAGIC was shown to be usable for creating gestures that are internally consistent and externally differentiable, and that have a low rate of false positives with respect to everyday life. I also identified strategies utilized by gesture designers to make their gestures more memorable and less likely to be accidentally activated by motions made by people during normal activities.

This dissertation can act as a starting point for researchers interested in mobile technology used in non-traditional situations, and provides new ways to think about interaction. The specific interaction techniques presented herein can be extended in a variety of ways to create new interfaces for mobile devices, and the idea of microinteractions provides a framework to use for thinking about devices in mobile situations.

REFERENCES

[1] Alpern, M. and Minardo, K., "Developing a car gesture interface for use as a secondary task," in *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, Apr 2003.

[2] Ardizzone, E., Chella, A., and Pirrone, R., "An architecture for automatic gesture analysis," in *Proceedings of the working conference on Advanced visual interfaces (AVI)*, May 2000.

[3] Ashbrook, D., Auxier, J., Gandy, M., and Starner, T., "Experiments in interaction between wearable and environmental infrastructure using the gesture pendant," in *Proceedings of Human Computer Interaction International (HCII) Workshop on Wearable Computing*, p. 5, Apr 2001.

[4] Ashbrook, D., Clawson, J., Lyons, K., Patel, N., and Starner, T., "Quickdraw: The impact of mobility and on-body placement on device access time," in *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, 2008.

[5] Ashbrook, D., Lyons, K., and Starner, T., "An investigation into round touchscreen wristwatch interaction," in *Proceedings of the ACM International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI)*, p. 4, May 2008.

[6] Baudisch, P. and Chu, G., "Back-of-device interaction allows creating very small touch devices," in *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, pp. 1923–1932, Jan 2009.

[7] Blaskó, G. B., *Cursorless Interaction Techniques for Wearable and Mobile Computing*. PhD thesis, Columbia University, Apr 2007.

[8] Bolt, R., ""put-that-there": Voice and gesture at the graphics interface," in *Proceedings of the 7th annual conference on Computer Graphics and Interactive Techniques*, Jan 1980.

[9] Bolt, R. and Herranz, E., "Two-handed gesture in multi-modal natural dialog," in *Proceedings of the ACM symposium on User interface software and technology (UIST)*, Dec 1992.

[10] Brashear, H., Park, K.-H., Lee, S., Henderson, V., Hamilton, H., and Starner, T., "American sign language recognition in game development for deaf children," in *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, pp. 79–86, 2006.

[11] Brown, L. M., Sellen, A., Krishna, R., and Harper, R., "Exploring the potential of audio-tactile messaging for remote interpersonal communication," in *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, pp. 1527–1530, Jan 2009.

[12] Cadoz, C., *Les réalités virtuelles*. Flammarion, 1994.

[13] Cadoz, C. and Wanderley, M., "Gesture-music," *Trends in Gestural Control of Music*, Jan 2000.

[14] Callahan, J., Hopkins, D., Weiser, M., and Shneiderman, B., "An empirical comparison of pie vs. linear menus," in *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, 1988.

[15] Chin, J., Diehl, V., and Norman, K., "Development of an instrument measuring user satisfaction of the human-computer interface," *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems*, May 1988.

[16] Chipchase, J., Persson, P., Piippo, P., Aarras, M., and Yamamoto, T., "Mobile essentials: field study and concepting," in *Designing for User eXperience*, 2005.

[17] Costanza, E., Inverso, S., Allen, R., and Maes, P., "Intimate interfaces in action: assessing the usability and subtlety of emg-based motionless gestures," in *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, Apr 2007.

[18] Cui, Y., Chipchase, J., and Ichikawa, F., "A cross culture study on phone carrying and physical personalization," in *Proceedings of Human Computer Interaction International (HCII)*, pp. 483–492, Jan 2007.

[19] Dannenberg, R. and Amon, D., "A gesture based user interface prototyping system," in *Proceedings of the ACM symposium on User interface software and technology (UIST)*, Nov 1989.

[20] Dey, A., Hamid, R., Beckmann, C., Li, I., and Hsu, D., "a cappella: programming by demonstration of context-aware applications," *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 33–40, Apr 2004.

[21] Deyle, T., Palinko, S., Poole, E. S., and Starner, T., "Hambone: A bioacoustic gesture interface," *International Symposium on Wearable Computers*, 2007.

[22] Fails, J. and Olsen, D., "A design tool for camera-based interaction," *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, Apr 2003.

[23] Froehlich, J., Wobbrock, J., and Kane, S., "Barrier pointing: using physical edges to assist target acquisition on mobile device touch screens," in *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, 2007.

[24] Fu, A. W.-C., Keogh, E., Lau, L. Y. H., Ratanamahatana, C. A., and Wong, R. C.-W., "Scaling and time warping in time series querying," *The VLDB Journal*, vol. 17, pp. 899–921, Jul 2008.

[25] Gábor Blaskó, *Cursorless Interaction Techniques for Wearable and Mobile Computing*. PhD thesis, Columbia University, 2007.

[26] Gábor Blaskó and Feiner, S., "Evaluation of an eyes-free cursorless numeric entry system for wearable computers," in *Proceedings of the IEEE International Symposium on Wearable Computers (ISWC)*, (Montreux, Switzerland), pp. 21–28, 2006.

[27] Gellersen, H., Schmidt, A., Beigl, M., and Thate, O., "Developing user interfaces for wearable computers: Don't stop to point and click," in *International Workshop on Interactive Applications of Mobile Computing*, 2000.

[28] Guimbretiére, F. and Winograd, T., "Flowmenu: combining command, text, and data entry," in *Proceedings of UIST*, 2000.

[29] Guo, C. and Sharlin, E., "Exploring the use of tangible user interfaces for human-robot interaction: a comparative study," *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, Apr 2008.

[30] Hartmann, B., Abdulla, L., Mittal, M., and Klemmer, S., "Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition," *CHI '07: Proceeding of the twenty-fifth annual SIGCHI conference on Human factors in computing systems*, pp. 1–10, Jan 2007.

[31] Hauptmann, A., "Speech and gestures for graphic image manipulation," *CHI '89: Proceedings of the SIGCHI conference on Human factors in computing systems: Wings for the mind*, Mar 1989.

[32] Hutchins, E., Hollan, J., and Norman, D., "Direct manipulation interfaces," *Human-Computer Interaction*, vol. 1, no. 4, pp. 311–338, 1985.

[33] Istance, H., Bates, R., Hyrskykari, A., and Vickers, S., "Snap clutch, a moded approach to solving the midas touch problem," *ETRA '08: Proceedings of the 2008 symposium on Eye tracking research & applications*, Mar 2008.

[34] Kallio, S., Kela, J., Mäntyjärvi, J., and Plomp, J., "Visualization of hand gestures for pervasive computing environments," *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, May 2006.

[35] Karam, M. and m schraefel, "Investigating user tolerance for errors in vision-enabled gesture-based interactions," *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, May 2006.

[36] Kavakli, M., Taylor, M., and Trapeznikov, A., "Designing in virtual reality (desire): a gesture-based interface," *DIMEA '07: Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts*, Sep 2007.

[37] Keates, S. and Robinson, P., "The use of gestures in multimodal input," *Assets '98: Proceedings of the third international ACM conference on Assistive technologies*, Jan 1998.

[38] Kela, J., Korpipää, P., Mäntyjärvi, J., Kallio, S., Savino, G., Jozzo, L., and Marca, D., "Accelerometer-based gesture control for a design environment," *Personal and Ubiquitous Computing*, vol. 10, Jul 2006.

[39] Këpuska, V. and Klein, T., "A novel wake-up-word speech recognition system, wake-up-word recognition task, technology and evaluation," *Nonlinear Analysis: Theory, Methods & Applications*, vol. In press, 2009.

[40] Kim, J., He, J., Lyons, K., and Starner, T., "The gesture watch: A wireless contact-free gesture based wrist interface," *International Symposium on Wearable Computers (ISWC)*, Jan 2007.

[41] Klemmer, S., Sinha, A., Chen, J., Landay, J., Aboobaker, N., and Wang, A., "Suede: a wizard of oz prototyping tool for speech user interfaces," *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, Nov 2000.

[42] Ko, M. H., West, G., Venkatesh, S., and Kumar, M., "Using dynamic time warping for online temporal fusion in multisensor systems," *Information Fusion*, vol. 9, no. 3, pp. 370–388, 2008.

[43] Kratz, S. and Ballagas, R., "Unravelling seams: Improving mobile gesture recognition with visual feedback techniques," *CHI '09: Proceeding of the twenty-seventh annual SIGCHI conference on Human factors in computing systems*, pp. 1–4, Jan 2009.

[44] Kristoffersen, S. and Ljungberg, F., ""Making place" to make IT work: empirical explorations of HCI for mobile CSCW," in *GROUP '99: Proceedings of the international ACM SIGGROUP conference on Supporting group work*, (New York, NY, USA), pp. 276–285, ACM, 1999.

[45] Krum, D., Omoteso, O., Ribarsky, W., Starner, T., and Hodges, L., "Speech and gesture multimodal control of a whole earth 3d visualization environment," *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*, May 2002.

[46] Kurtenbach, G., *The Design and Evaluation of Marking Menus.* PhD thesis, University of Toronto, 1993.

[47] LaViola, J., Feliz, D., Keefe, D., and Zeleznik, R., "Hands-free multi-scale navigation in virtual environments," *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, Mar 2001.

[48] Lee, C., Ghyme, S., Park, C., and Wohn, K., "The control of avatar motion using hand gesture," *VRST '98: Proceedings of the ACM symposium on Virtual reality software and technology*, Nov 1998.

[49] Lenman, S., Bretzner, L., and Thuresson, B., "Using marking menus to develop command sets for computer vision based hand gesture interfaces," *NordiCHI '02: Proceedings of the second Nordic conference on Human-computer interaction*, Oct 2002.

[50] Linjama, J., Korpipää, P., Kela, J., and Rantakokko, T., "Actioncube: a tangible mobile gesture interaction tutorial," *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, Feb 2008.

[51] Liu, J., Wang, Z., Zhong, L., Wickramasuriya, J., and Vasudevan, V., "uwave: Accelerometer-based personalized gesture recognition and its applications," *IEEE International Conference on Pervasive Computing and Communications*, pp. 1–9, 2009.

[52] Long, A., Landay, J., and Rowe, L., “”those look similar!” issues in automating gesture design advice,” *PUI '01: Proceedings of the 2001 workshop on Perceptive user interfaces*, Nov 2001.

[53] Long, A., Jr, Landay, J., and Rowe, L., “Implications for a gesture design tool,” *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, pp. 40–47, May 1999.

[54] Long, C. A., *Quill: a Gesture Design Tool for Pen-based User Interfaces.* PhD thesis, University of California, Berkeley, Jun 2001.

[55] Lyons, K., Brashear, H., Westeyn, T., and Kim, J., “Gart: The gesture and activity recognition toolkit,” *Human-Computer Interaction International.*, pp. 718–727, Jan 2007.

[56] Lyons, K., Skeels, C., Starner, T., Snoeck, C., Wong, B., and Ashbrook, D., “Augmenting conversations using dual-purpose speech,” *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, Oct 2004.

[57] Mankoff, J. and Abowd, G. D., “Cirrin: a word-level unistroke keyboard for pen input,” in *Proceedings of UIST*, 1998.

[58] Mäntyjärvi, J., Kela, J., Korpipää, P., and Kallio, S., “Enabling fast and effortless customisation in accelerometer based gesture interaction,” *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, Oct 2004.

[59] Maynes-Aminzade, D., Winograd, T., and Igarashi, T., “Eyepatch: prototyping camera-based interaction through examples,” *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, Oct 2007.

[60] McNeill, D., *Hand and Mind: What Gestures Reveal about Thought.* University Of Chicago Press, 1996.

[61] Miller, R., “Response time in man-computer conversational transactions,” *AFIPS '68 (Fall, part I): Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, Dec 1968.

[62] Morrel-Samuels, P., “Clarifying the distinction between lexical and gestural commands,” *International Journal of Man-Machine Studies*, vol. 32, no. 5, pp. 581–590, 1990.

[63] Moscovich, T. and Hughes, J., “Navigating documents with the virtual scroll ring,” in *Proceedings of UIST*, 2004.

[64] Mueen, A., Keogh, E., Zhu, Q., Cash, S., and Westover, B., “Exact discovery of time series motifs,” *SIAM International Conference on Data Mining*, 2009.

[65] Myers, B., Hudson, S. E., and Pausch, R., “Past, present, and future of user interface software tools,” *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 7, no. 1, pp. 3–28, 2000.

[66] Nielsen, J., "Iterative user-interface design," *IEEE Computer*, vol. 26, no. 11, pp. 32–41, 1993.

[67] Nielsen, M., Störring, M., Moeslund, T., and Granum, E., "A procedure for developing intuitive and ergonomic gesture interfaces for hci," *5th International Gesture Workshop*, Jan 2003.

[68] Oakley, I. and Park, J., "A motion-based marking menu system," *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, Apr 2007.

[69] Oulasvirta, A., Tamminen, S., Roto, V., and Kuorelahti, J., "Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile hci," *Conference on Human Factors in Computing Systems*, pp. 919–928, 2005.

[70] Oulasvirta, A., "The fragmentation of attention in mobile interaction, and what to do with it," *ACM interactions*, vol. 12, no. 6, pp. 16–18, 2005.

[71] Patel, K., Fogarty, J., Landay, J., and Harrison, B., "Examining difficulties software developers encounter in the adoption of statistical machine learning," *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 1563–1566, Apr 2008.

[72] Patel, S., Kientz, J., Hayes, G., Bhat, S., and Abowd, G., "Farther than you may think: An empirical investigation of the proximity of users to their mobile phones.," in *Ubicomp*, pp. 123–140, 2006.

[73] Pausch, R. and Williams, R., "Tailor: creating custom user interfaces based on gesture," *UIST '90: Proceedings of the 3rd annual ACM SIGGRAPH symposium on User interface software and technology*, Aug 1990.

[74] Payne, J., Keir, P., Elgoyhen, J., McLundie, M., Naef, M., Horner, M., and Anderson, P., "Gameplay issues in the design of spatial 3d gestures for video games.," *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, Apr 2006.

[75] Perlin, K., "Quikwriting: continuous stylus-based text entry," in *Proceedings of UIST*, 1998.

[76] Pirhonen, A., Brewster, S., and Holguin, C., "Gestural and audio metaphors as a means of control for mobile devices," *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, Apr 2002.

[77] Prekopcsák, Z., "Accelerometer based real-time gesture recognition," *Proceedings of the 12th International Student Conference on Electrical Engineering*, pp. 1–5, Mar 2008.

[78] Rachovides, D., Walkerdine, J., and Phillips, P., "The conductor interaction method," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP*, vol. 3, Dec 2007.

[79] Raghunath, M. T. and Narayanaswami, C., "User interfaces for applications on a wrist watch," *Personal and Ubiquitous Computing*, vol. 6, pp. 17–30, February 2002.

[80] Rekimoto, J., "Gesturewrist and gesturepad: Unobtrusive wearable interaction devices," *Proc. ISWC'01, Fifth Intl Symposium on Wearable Computers*, pp. 87–94, 2001.

[81] Robbe, S., "An empirical study of speech and gesture interaction: toward the definition of ergonomic design guidelines," *CHI '98: CHI 98 conference summary on Human factors in computing systems*, Apr 1998.

[82] Romero, M., Summet, J., Stasko, J., and Abowd, G., "Viz-a-vis: Toward visualizing video through computer vision," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 14, pp. 1261 – 1268, Nov 2008.

[83] Ronkainen, S., Häkkilä, J., Kaleva, S., Colley, A., and Linjama, J., "Tap input as an embedded interaction method for mobile devices," *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*, Feb 2007.

[84] Rossini, N., *Participatory Design: Principles and Practices.* Hillsdale, NJ: Lawrence Erlbaum, 1993.

[85] Rubegni, E., Brunk, J., Caporali, M., and Rizzo, A., "Wi-roni: a gesture tangible interface for experiencing internet content in public spaces," *TMR '07: Proceedings of the 2007 workshop on Tagging, mining and retrieval of human related activity information*, Nov 2007.

[86] Saponas, T., Tan, D., Morris, D., Balakrishnan, R., Turner, J., and Landay, J., "Enabling always-available input with muscle-computer interfaces," in *Proceedings of the ACM symposium on User interface software and technology (UIST)*, Oct 2009.

[87] Schmidt, R., Zelaznik, H., Hawkins, B., Frank, J., and Quinn, J., "Motor-output variability: A theory for the accuracy of rapid motor acts," *Psychological Review*, vol. 86, no. 5, 1979.

[88] Segen, J. and Kumar, S., "Gesture vr: vision-based 3d hand interace for spatial interaction," *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, Sep 1998.

[89] Smith, G. M. and m. c. schraefel, "The radial scroll tool: scrolling support for stylus-or touch-based document navigation," in *Proceedings of UIST*, 2004.

[90] Smith, G., m. c. schraefel, and Baudisch, P., "Curve dial: eyes-free parameter entry for guis," in *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, 2005.

[91] Starner, T., Auxier, J., Ashbrook, D., and Gandy, M., "The gesture pendant: a self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring," *International Symposium on Wearable Computers (ISWC)*, pp. 87 – 94, Oct 2000.

[92] Starner, T., Auxier, J., Ashbrook, D., and Gandy, M., "The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring," in *Proceedings of the IEEE International Symposium on Wearable Computers (ISWC)*, (Atlanta, GA), 2000.

[93] Starner, T., Snoeck, C., Wong, B., and McGuire, R., "Use of mobile appointment scheduling devices," in *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, 2004.

[94] Starner, T., Snoeck, C. M., Wong, B., and McGuire, R. M., "Use of mobile appointment scheduling devices," *Human Factors in Computing Systems (CHI)*, pp. 1–4, Mar 2004.

[95] Strachan, S., Murray-Smith, R., and O'Modhrain, S., "Bodyspace: inferring body pose for natural control of a music player," *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, Apr 2007.

[96] Takahashi, T. and Kishino, F., "Hand gesture coding based on experiments using a hand gesture interface device," *ACM SIGCHI Bulletin*, vol. 23, Mar 1991.

[97] Talbot, J., Lee, B., Kapoor, A., and Tan, D., "Ensemblematrix: Interactive visualization to support machine learning with multiple classifiers," *CHI '09: Proceeding of the twenty-seventh annual SIGCHI conference on Human factors in computing systems*, pp. 1283–1292, Jan 2009.

[98] Tormene, P., Giorgino, T., Quaglini, S., and Stefanelli, M., "Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation," *Artificial Intelligence in Medicine*, vol. 45, no. 1, pp. 11–34, 2009.

[99] Vadas, K., Patel, N., Lyons, K., Starner, T., and Jacko, J., "Reading on-the-go: a comparison of audio and hand-held displays," in *MobileHCI*, 2006.

[100] Vogel, D. and Baudisch, P., "Shift: A technique for operating pen-based interfaces using touch," in *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, 2007.

[101] Westeyn, T., Brashear, H., Atrash, A., and Starner, T., "Georgia tech gesture toolkit: supporting experiments in gesture recognition," *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, Nov 2003.

[102] Westeyn, T. and Starner, T., "Recognizing song–based blink patterns: Applications for restricted and universal access," *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 717–722, Oct 2004.

[103] Wexelblat, A., "An approach to natural gesture in virtual environments," *ACM Transactions on Computer-Human Interaction (TOCHI*, vol. 2, Sep 1995.

[104] Wexelblat, A., "Gesture at the user interface: a chi '95 workshop," *ACM SIGCHI Bulletin*, vol. 28, Apr 1996.

[105] Witten, I. H. and Frank, E., *Data Mining: Practical Machine Learning Tools and Techniques.* San Francisco: Morgan Kaufmann, 2nd ed., 2005.

[106] Wobbrock, J., Cutrell, E., Harada, S., and MacKenzie, I., "An error model for pointing based on Fitts' law," in *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, 2008.

[107] Wobbrock, J., Morris, M. R., and Wilson, A. D., "User-defined gestures for surface computing," *CHI '09: Proceeding of the twenty-seventh annual SIGCHI conference on Human factors in computing systems*, pp. 1083–1092, Jan 2009.

[108] Wu, J., Pan, G., Zhang, D., Qi, G., and Li, S., "Gesture recognition with a 3-d accelerometer," *Ubiquitous Intelligence and Computing*, vol. 5585, Jan 2009.

[109] Zhao, S., Dragicevic, P., Chignell, M., Balakrishnan, R., and Baudisch, P., "Earpod: eyes-free menu selection using touch input and reactive audio feedback," in *Proceedings of SIGCHI conference on Human Factors in Computing Systems (CHI)*, 2007.

[110] Zimmerman, T., Lanier, J., Blanchard, C., Bryson, S., and Harvill, Y., "A hand gesture interface device," *CHI '87: Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface*, May 1987.

APPENDIX A

MAGIC DEMOGRAPHIC SURVEY

The next page is the demographic survey that participants were requested to fill out before the MAGIC evaluation (described in Chapter 6) began.

About You

This questionnaire asks you about yourself. All information will be kept confidential.

Sex:	Male / Female
Age:	_______
Do you wear a watch?	Yes / No / Sometimes
If so, on which wrist?	Right / Left

How much experience do you have with the Nintendo Wii game system?

never played — I play it every day

1	2	3	4	5	6	7	8	9

How much experience do you have with motion-sensing mobile phones such as the Apple iPhone or the T-Mobile G1?

Never used	Have tried, but don't own	Owned < 6 months	Owned 6 mos.–1 year	Owned 1 year–1.5 years	Owned > 1.5 years

If you own such a mobile phone, which one? ____

How much experience do you have with user-centered design?

completely inexperienced — very experienced

1	2	3	4	5	6	7	8	9

How experienced are you in designing user interfaces?

completely inexperienced — very experienced

1	2	3	4	5	6	7	8	9

APPENDIX B

MAGIC POST-STUDY QUESTIONNAIRE

The next 9 pages consist of the questionnaire that participants were requested to complete after the study (described in Chapter 6) ended.

Experiment Questionnaire

We would like you to tell us about your experiences with the MAGIC software. Please be honest —we really want to know what you think!

Please select the numbers which most appropriately reflect your impressions about the software and the experimental task. *Note: NA = Not Applicable.*

If you have strong feelings about any part of the software or about the experiment, please write them down in the space provided at the bottom of each page. If you have suggestions on how to improve the MAGIC software, please write those down as well. Your ratings and anything you write will be kept confidential.

Please enter any general comments about the experiment below:

Overall Reactions to the Software

terrible — wonderful

1	2	3	4	5	6	7	8	9	NA

frustrating — satisfying

1	2	3	4	5	6	7	8	9	NA

dull — stimulating

1	2	3	4	5	6	7	8	9	NA

difficult — easy

1	2	3	4	5	6	7	8	9	NA

inadequate power — adequate power

1	2	3	4	5	6	7	8	9	NA

rigid — flexible

1	2	3	4	5	6	7	8	9	NA

Please write any comments about the software below:

Information Presentation

Screen layouts were helpful:

never								always	
1	2	3	4	5	6	7	8	9	NA

Amount of information that can be displayed on the screen:

inadequate								adequate	
1	2	3	4	5	6	7	8	9	NA

Arrangement of information on the screen:

illogical								logical	
1	2	3	4	5	6	7	8	9	NA

Sequence of screens (Gesture Creation/Gesture Testing/Everyday Gesture Library):

confusing								clear	
1	2	3	4	5	6	7	8	9	NA

Progression of work-related tasks:

confusing								clearly marked	
1	2	3	4	5	6	7	8	9	NA

Please write any comments about the presentation of information in the software below:

Terminology and System Information

Use of terminology throughout system:

inconsistent consistent

1	2	3	4	5	6	7	8	9	NA

Computer keeps you informed about what it is doing:

never always

1	2	3	4	5	6	7	8	9	NA

Performing an operation leads to a predictable result:

never always

1	2	3	4	5	6	7	8	9	NA

Information about quality of gestures:

confusing clear

1	2	3	4	5	6	7	8	9	NA

Please write any comments about the presentation of information in the software below:

Learning

Learning to operate the system:

difficult easy

1	2	3	4	5	6	7	8	9	NA

Time to learn to use the system:

slow fast

1	2	3	4	5	6	7	8	9	NA

Exploration of features by trial and error:

discouraging encouraging

1	2	3	4	5	6	7	8	9	NA

Tasks can be performed in a straight-forward manner:

never always

1	2	3	4	5	6	7	8	9	NA

Steps to complete a task follow a logical sequence:

never always

1	2	3	4	5	6	7	8	9	NA

Please write any comments about the learnability of the software below:

System Capabilities

System speed:

too slow fast enough

1	2	3	4	5	6	7	8	9	NA

Correcting your mistakes:

difficult easy

1	2	3	4	5	6	7	8	9	NA

Ease of operation depends on your level of experience:

never always

1	2	3	4	5	6	7	8	9	NA

Please write any comments about the capabilities of the software below:

Experiment

Match between designed gestures and intended purposes:

low high

1	2	3	4	5	6	7	8	9	NA

Ease for someone else to learn designed gestures:

low high

1	2	3	4	5	6	7	8	9	NA

I would like to own the uPod Touchless, to use with my designed gestures:

no way definitely

1	2	3	4	5	6	7	8	9	NA

Are there any other devices you'd like to control with gestures? If so, please briefly describe each device and what gestural control you would like for it to have:

Gesture Design page 1

We’re interested in your design choices; for each gesture, please write a short description of why you chose the particular gesture for the given functionality. If you had a strong impression of what the gesture should be, but ended up not using that idea, please write down the idea and why it didn’t work out. You can go back and watch your videos if you need a reminder of what the gestures looked like.

play/pause:

next track:

previous track:

next playlist:

Gesture Design page 2

We're interested in your design choices; for each gesture, please write a short description of why you chose the particular gesture for the given functionality. If you had a strong impression of what the gesture should be, but ended up not using that idea, please write down the idea and why it didn't work out. You can go back and watch your videos if you need a reminder of what the gestures looked like.

previous playlist:

volume up by 10%:

volume down by 10%:

shuffle:

APPENDIX C

MAGIC CONDITION A TASK SHEET

The next page is the handout given to participants in condition A describing the experimental task to be completed for the study described in Chapter 6.

The year is 2012. Pear Computer is working on their latest uPod product. In keeping with their well-known “less is more” strategy, the new “uPod Touchless” will be a wrist-wearable music player that is controlled through gestures. Rather than having to fumble for buttons, the user will control her music with simple wrist motions.

There is just one problem: early user testing has revealed that users’ everyday motions—such as waving hello to someone or eating a sandwich—tend to trigger undesired uPod functions. This is particularly a problem with the new “always on” wireless earbuds the uPod is supposed to come with: when a customer waves hello to a friend, accidentally starting her music playing, she can’t hear the friend’s response!

Your job, as an employee of the newly-formed Gesture Engineering Group at Pear, is to create a new set of potential gestures for the uPod Touchless. The goal is to create gestures that are easy to learn and remember, but that aren’t similar enough to users’ everyday lives to accidentally activate uPod functionality.

The gesture-activated uPod functions that management has decided to include are:

- play/pause
- next track
- previous track
- next playlist
- previous playlist
- volume up by 10%
- volume down by 10%
- shuffle

In order for a gesture to be good enough to include on the uPod, it must meet the following criteria:

1. The gesture must reliably activate the desired function.
2. Performing the gesture must not activate other functions.
3. The functionality associated with a gesture must not be activated by a user’s everyday movements.
4. The gesture should be easy to remember.
5. The gesture should be easy to perform.
6. The gesture should be socially acceptable.

Using MAGIC, you’ll create gestures for each of the above uPod functions, test them out, and try to determine if the gestures you create will occur in a user’s everyday life. When you’re finished, an Pear Gesture Test Engineer will evaluate your gesture set to see how well it met the above criteria.

APPENDIX D

MAGIC CONDITION B TASK SHEET

The next page is the handout given to participants in condition B describing the experimental task to be completed for the study described in Chapter 6.

The year is 2012. Pear Computer is working on their latest uPod product. In keeping with their well-known "less is more" strategy, the new "uPod Touchless" will be a wrist-wearable music player that is controlled through gestures. Rather than having to fumble for buttons, the user will control her music with simple wrist motions.

There is just one problem: early user testing has revealed that users' everyday motions—such as waving hello to someone or eating a sandwich—tend to trigger undesired uPod functions. This is particularly a problem with the new "always on" wireless earbuds the uPod is supposed to come with: when a customer waves hello to a friend, accidentally starting her music playing, she can't hear the friend's response!

Your job, as an employee of the newly-formed Gesture Engineering Group at Pear, is to create a new set of potential gestures for the uPod Touchless. The goal is to create gestures that are easy to learn and remember, but that aren't similar enough to users' everyday lives to accidentally activate uPod functionality.

The gesture-activated uPod functions that management has decided to include are:

- play/pause
- next track
- previous track
- next playlist
- previous playlist
- volume up by 10%
- volume down by 10%
- shuffle

In order for a gesture to be good enough to include on the uPod, it must meet the following criteria:

1. The gesture must reliably activate the desired function.
2. Performing the gesture must not activate other functions.
3. The functionality associated with a gesture must not be activated by a user's everyday movements.
4. The gesture should be easy to remember.
5. The gesture should be easy to perform.
6. The gesture should be socially acceptable.

Using MAGIC, you'll create gestures for each of the above uPod functions, test them out, and use the EGL to try to determine if the gestures you create will occur in a user's everyday life. When you're finished, an Pear Gesture Test Engineer will evaluate your gesture set to see how well it met the above criteria.

APPENDIX E

MAGIC CONDITION B TUTORIAL

The next 26 pages consist of the tutorial provided to condition B and C participants; the tutorial for condition A participants was identical, but with all references to the EGL removed.

MAGIC Tutorial

1

Using MAGIC

MAGIC—the **M**ultiple **A**ction **G**esture **I**nterface **C**reation tool—is a piece of software used to help designers create gestures.

When we talk about *gestures*, we're talking about gestures made with your body, in three dimensions. This is different from gestures made on a touchscreen or other surface; the movements made with the Nintendo Wii controller are examples of the kinds of gestures we're concerned with.

Gestures can be used for a lot of different things, from video games to controlling robotic prosthetics. Normally, one gesture (like a cutting motion with the Wii remote) will activate one particular function (like swinging a sword in a video game).

One reason that gesture-based interfaces are currently mostly used for games is that they can be difficult to design. MAGIC was created to help.

Phases of Gesture Design

MAGIC supports two main phases of gesture design: *creation* and *testing*.

In the *creation* phase, you, as the designer, come up with with gestures that might be good to use for the functions you want to control. You might decide that a cutting motion is just the thing for your sword-fighting game. You next create some examples of that gesture, training the system to recognize the gesture.

In the *testing* phase, you test out your gestures. You make some motions that are like your gestures, and some that aren't, to make sure that each gesture is recognized correctly, and that movements that aren't intentional gestures aren't recognized.

This tutorial will help you learn how to use MAGIC to design gestures.

Eat Your Vegetables!

For the purposes of this tutorial, we'll consider this Japanese vegetable vending machine.

We want to make it gesture activated, so you can wave your hand and automatically dispense a vegetable!

For example, you might have a gesture to get out a Pickle, or a Parsnip, or a Turnip!

We have to be sure to think about the user: will the gesture be easy to remember and perform? And will the gesture be unique enough to avoid accidentally dispensing vegetables when someone waves to a friend?

To prevent people from getting vegetables for free, we're going to put the gesture-sensing device into a handy wristwatch form-factor. This way we can only give vegetables to people who have accounts with the farmer.

The amazing new VegiWatch™ uses an accelerometer to sense gestures. You'll learn about how it works on the next page.

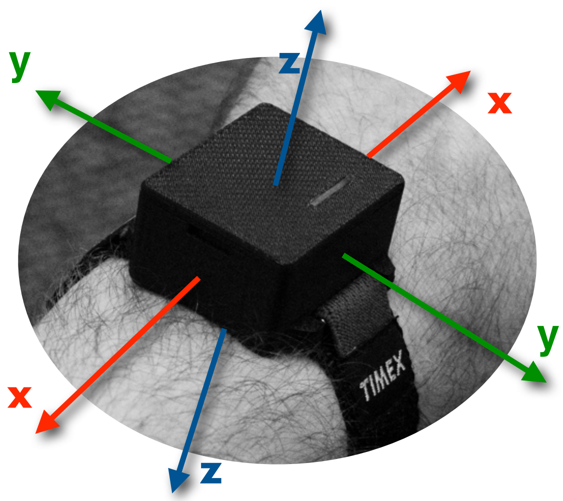
Accelerometers

In this tutorial, and in the rest of the experiment, you'll use an accelerometer for designing and testing your gestures.

An *accelerometer* is a device that senses acceleration. Acceleration simply means changes in speed.

The accelerometer that you'll use during this experiment can sense acceleration in three directions; we'll call them **x**, **y** and **z**.

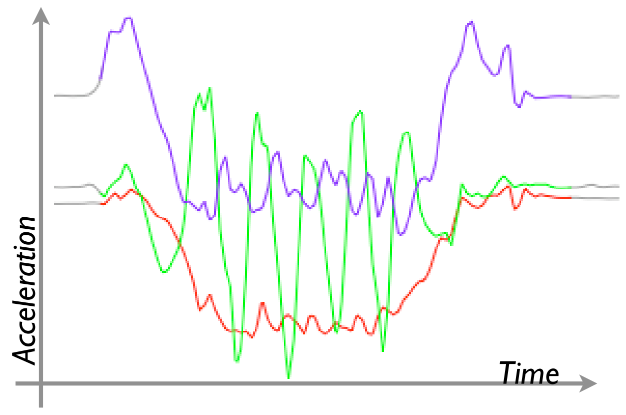
In the picture at right, the box shows the accelerometer, in the way that it is mounted on your wrist. Each colored line shows a direction that the accelerometer can sense in.



At bottom right is a typical view that you'll see in the MAGIC software. It shows a recorded gesture. Time is shown on the horizontal axis, and the magnitude of acceleration is shown on the vertical axis.

Each colored line in the plot corresponds to one of the directions sensed by the accelerometer. Plots like this can be difficult to interpret, but this is the same information that the computer has. It is not possible to take acceleration values and figure out the movements that lead to those values.

This is because two different movements can lead to very similar acceleration plots. Imagine quickly moving your arm from left to right. Now imagine rotating your wrist by 90° and moving it quickly up. The plot produced will be the same with each!



Gesture Recognition

In MAGIC, gesture recognition works through a process called example matching. This is a method of taking some input, called the *candidate*, and trying to figure out which gesture it most closely matches, if any.

Let's take an example. Say you're building a device that will detect different kinds of waves. You want it to be able to tell the difference between a friendly wave, a "parade wave", and the "thumbs up" gesture.

First, you would record a bunch of examples for each type of wave. Then, given an unknown candidate, the system would try to figure out which of your examples is the closest match.

This happens through a process called Dynamic Time Warping (DTW). This works by looking for similar parts of two signals. For example, say your candidate gesture looks like the red line at top right. The blue line represents one of the examples you recorded for "Hello!".

The two signals, red and blue, look pretty similar. But they're not exactly the same—the hump happens later in the red signal than in the blue.

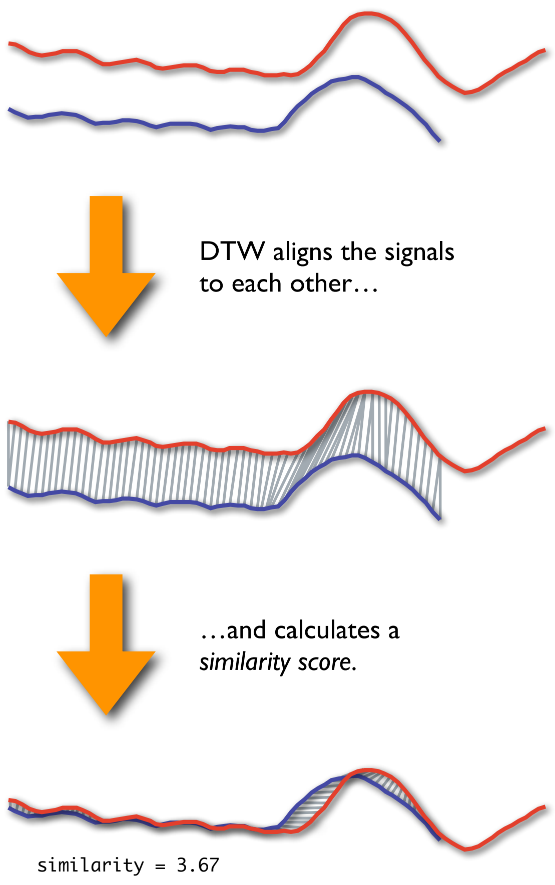
This is where DTW comes in! DTW works by looking for parts of two signals that are most similar.

In the middle picture, you can see the results. The grey lines connect parts of the two signals that are most similar to each other, so the two humps get connected at their peaks.

But how can we be sure this is the best match? What if there's another example that is better? To find out, we can assign a similarity score to the example gesture. The more similar the example gesture is to the candidate, the more like each other they are.

To visualize the calculation of the similarity score, look at the bottom picture. The two signals have been put on top of each other, and the grey lines have moved along with them. Now, to get the similarity score, we just have to add up the lengths of all of the grey lines!

If the blue and red signals were exactly the same, the grey lines would all be of length zero, so the similarity score would be zero—the best. In this case, they're close, but not exact. You'll see the similarity score used several places in MAGIC.


DTW aligns the signals to each other…
…and calculates a *similarity score.*
similarity = 3.67

Introduction to MAGIC

Gesture Creation Tab

MAGIC main screen

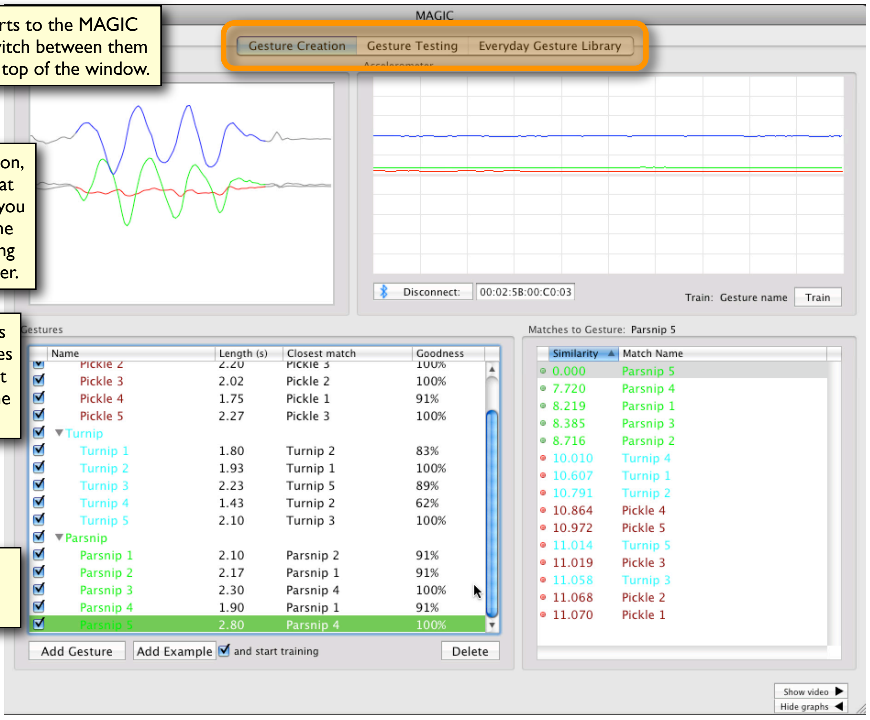
This image shows the main screen of MAGIC, mid-project.

There are three parts to the MAGIC interface. You can switch between them using the tabs at the top of the window.

The first tab, Gesture Creation, lets you create gestures that you want to try out. It gives you lots of information about the gestures you create, including how they relate to each other.

Gestures are like containers that hold *examples*. Examples are similar movements that should be considered by the system to be the same.

For example, the Parsnip gesture is made up of five different examples.

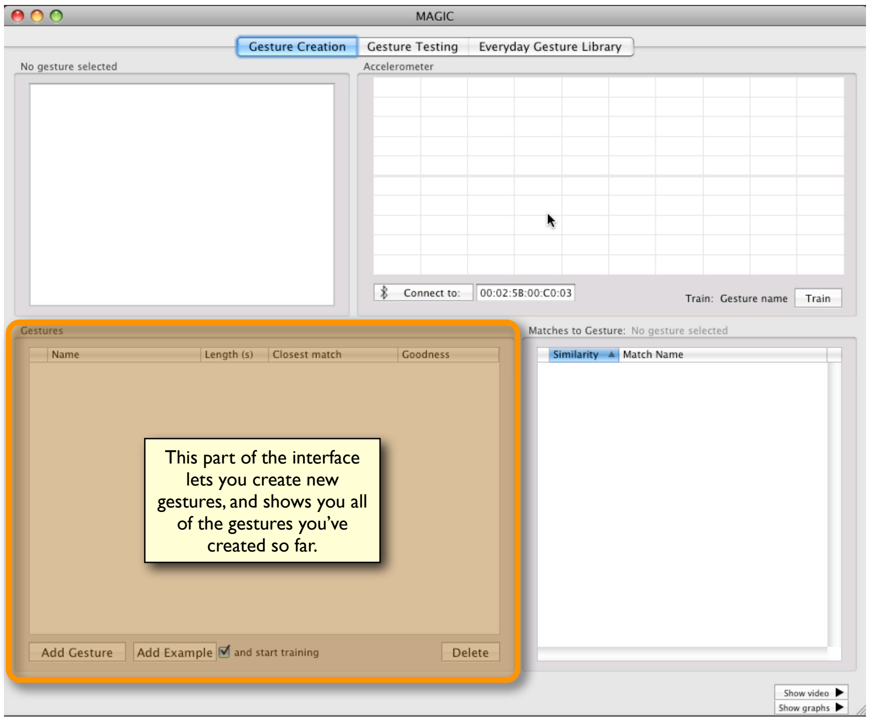


Name	Length (s)	Closest match	Goodness
Pickle 2	2.20	Pickle 3	100%
Pickle 3	2.02	Pickle 2	100%
Pickle 4	1.75	Pickle 1	91%
Pickle 5	2.27	Pickle 3	100%
▼ Turnip			
Turnip 1	1.80	Turnip 2	83%
Turnip 2	1.93	Turnip 1	100%
Turnip 3	2.23	Turnip 5	89%
Turnip 4	1.43	Turnip 2	62%
Turnip 5	2.10	Turnip 3	100%
▼ Parsnip			
Parsnip 1	2.10	Parsnip 2	91%
Parsnip 2	2.17	Parsnip 1	91%
Parsnip 3	2.30	Parsnip 4	100%
Parsnip 4	1.90	Parsnip 1	91%
Parsnip 5	2.80	Parsnip 4	100%

Similarity ▲	Match Name
0.000	Parsnip 5
7.720	Parsnip 4
8.219	Parsnip 1
8.385	Parsnip 3
8.716	Parsnip 2
10.010	Turnip 4
10.607	Turnip 1
10.791	Turnip 2
10.864	Pickle 4
10.972	Pickle 5
11.014	Turnip 5
11.019	Pickle 3
11.058	Turnip 3
11.068	Pickle 2
11.070	Pickle 1

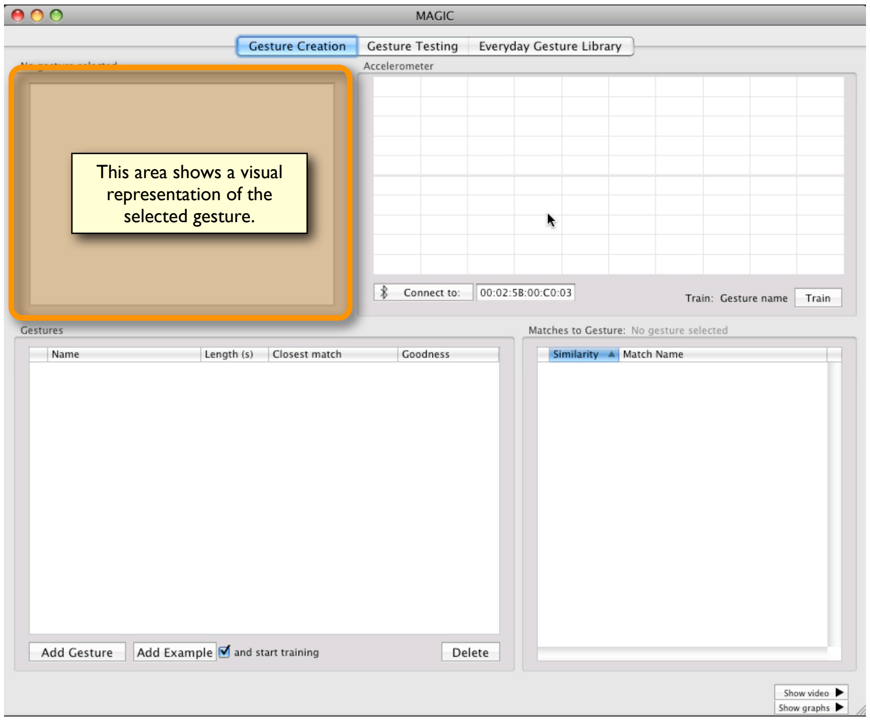
The next few pages in this tutorial will introduce the parts of the Gesture Creation tab.

MAGIC main screen

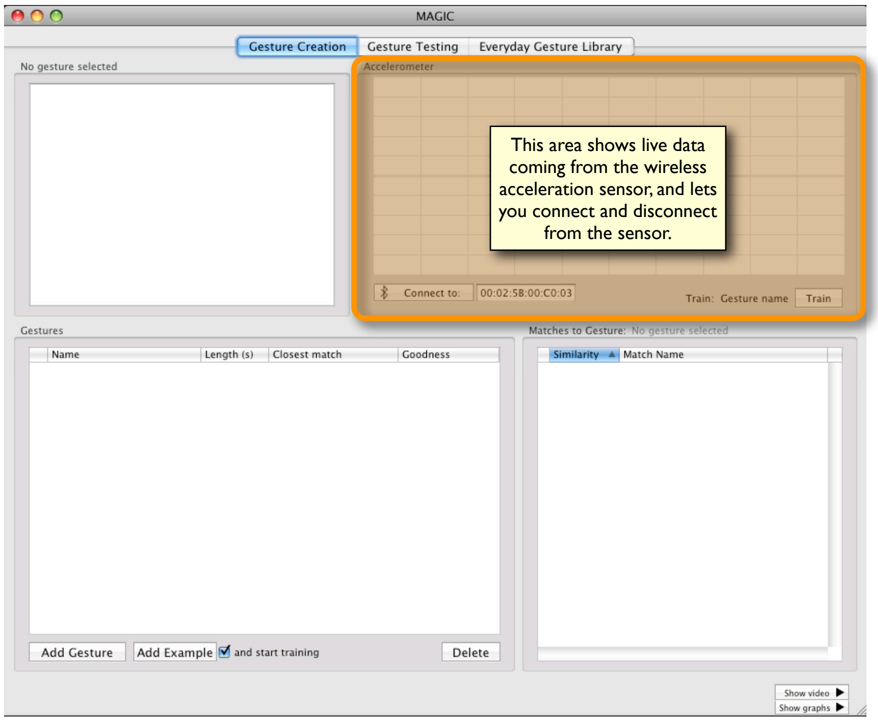


This part of the interface lets you create new gestures, and shows you all of the gestures you've created so far.

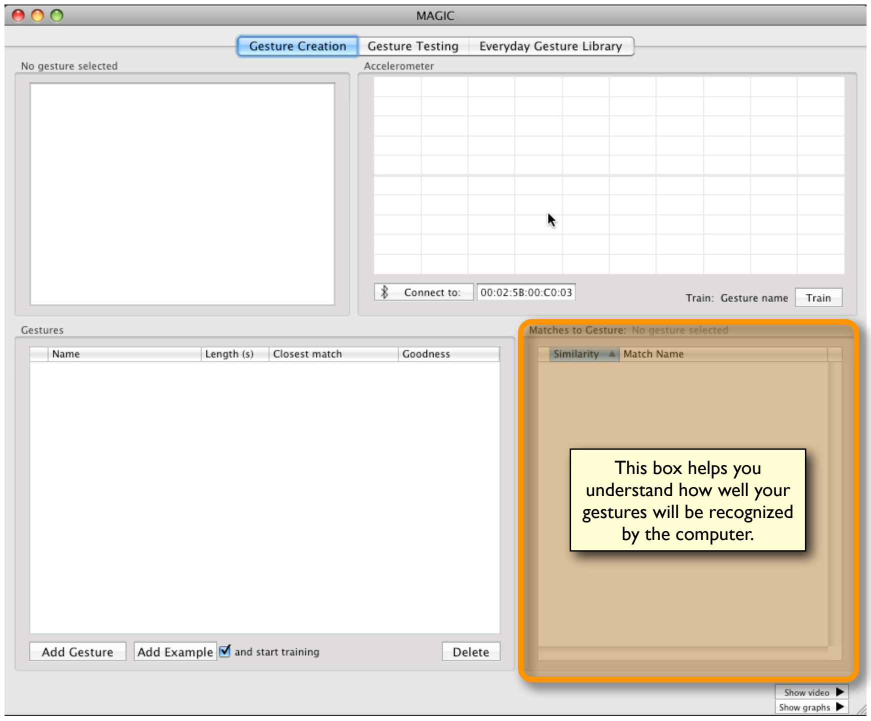
MAGIC main screen


This area shows a visual representation of the selected gesture.

MAGIC main screen

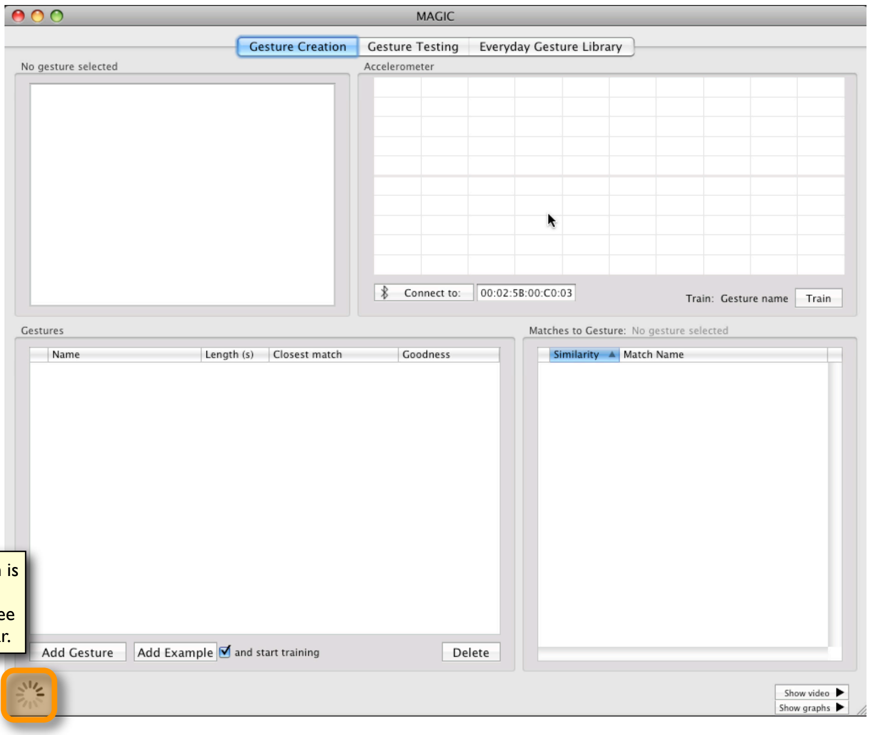

This area shows live data coming from the wireless acceleration sensor, and lets you connect and disconnect from the sensor.

MAGIC main screen



This box helps you understand how well your gestures will be recognized by the computer.

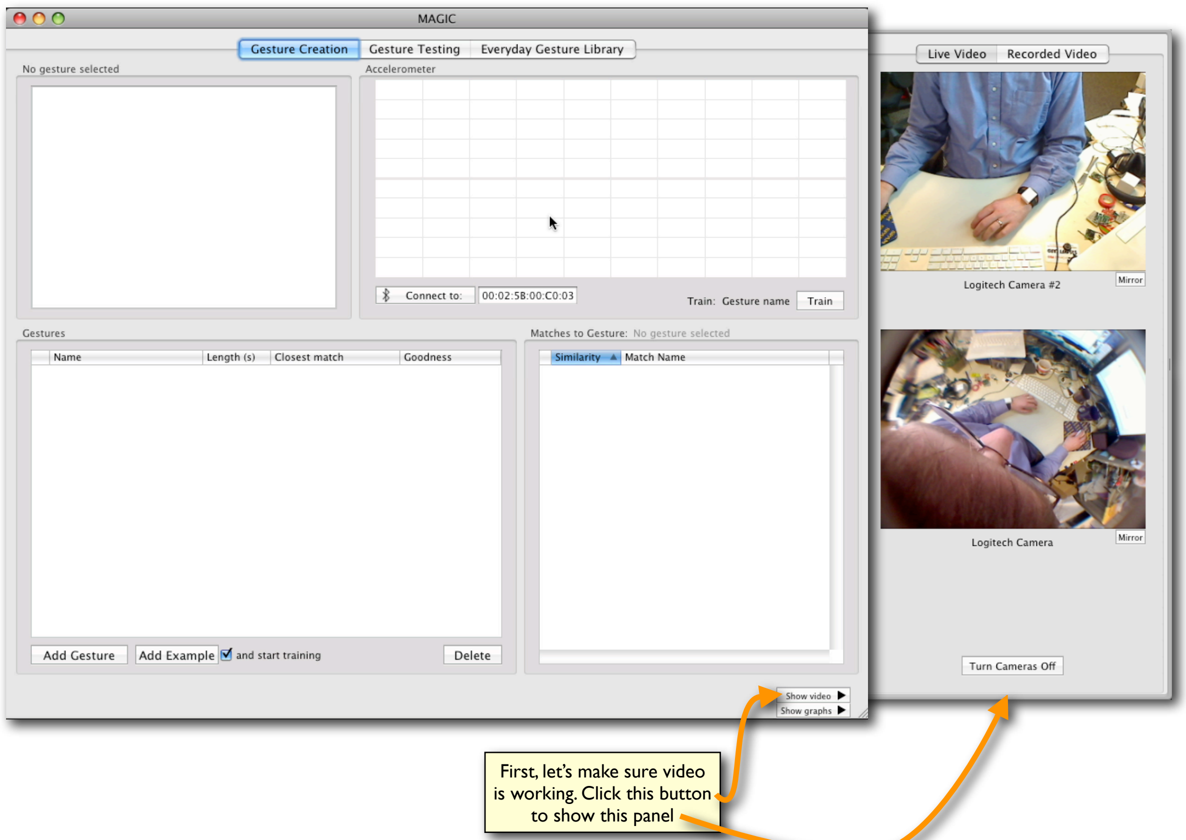
MAGIC main screen



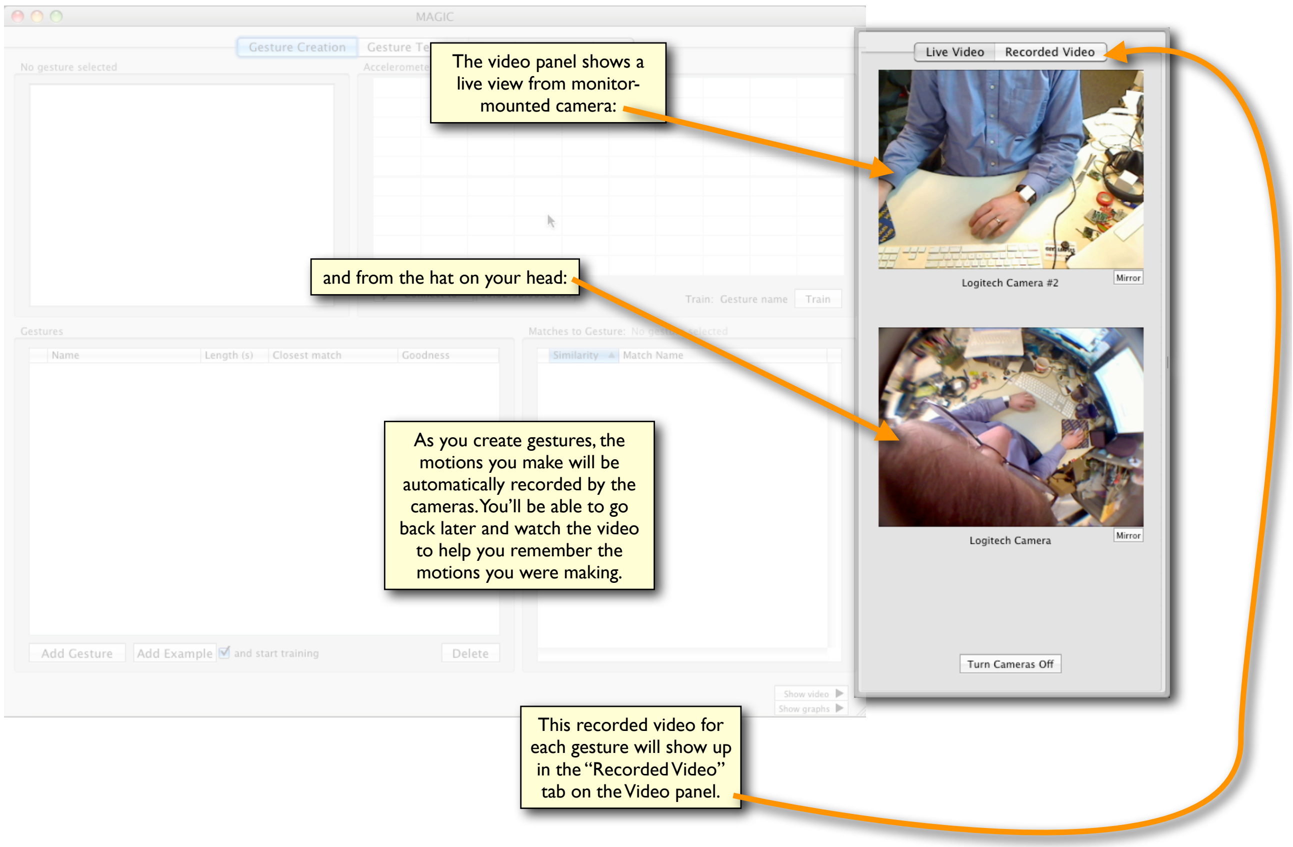
When the program is busy calculating something, you'll see this spinner appear.

Introduction to MAGIC

The gesture creation process



First, let's make sure video is working. Click this button to show this panel



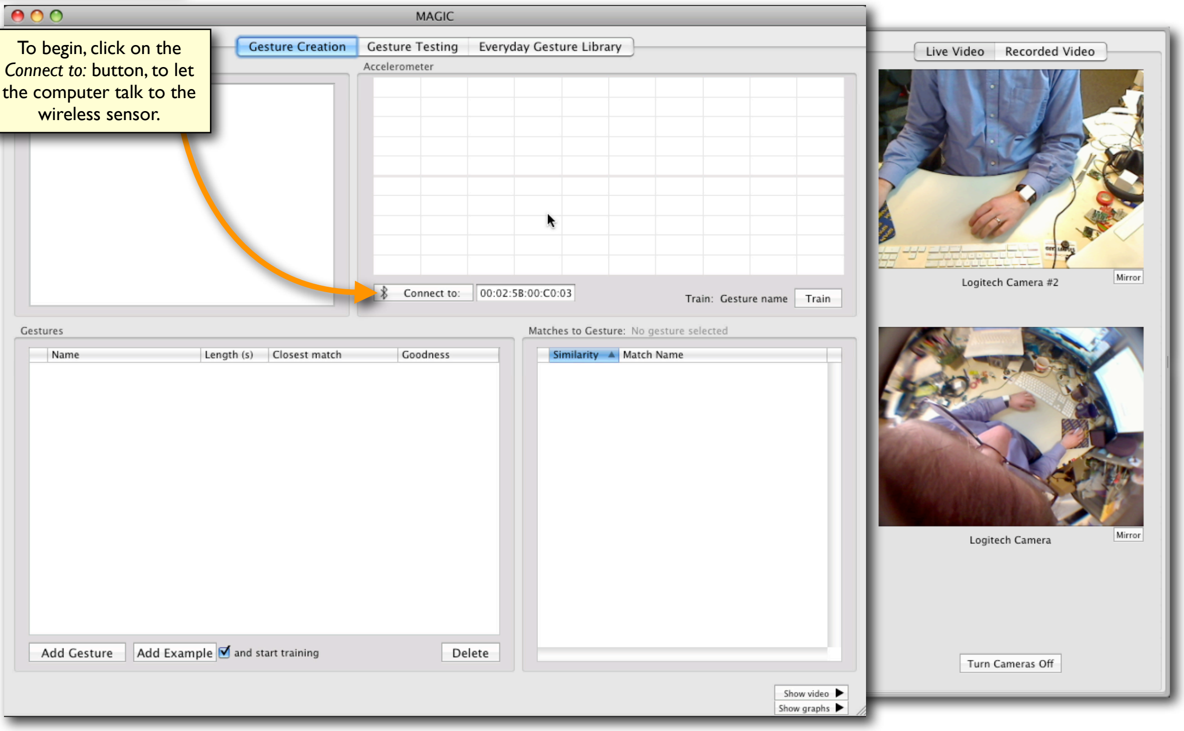
The video panel shows a live view from monitor-mounted camera:

and from the hat on your head:

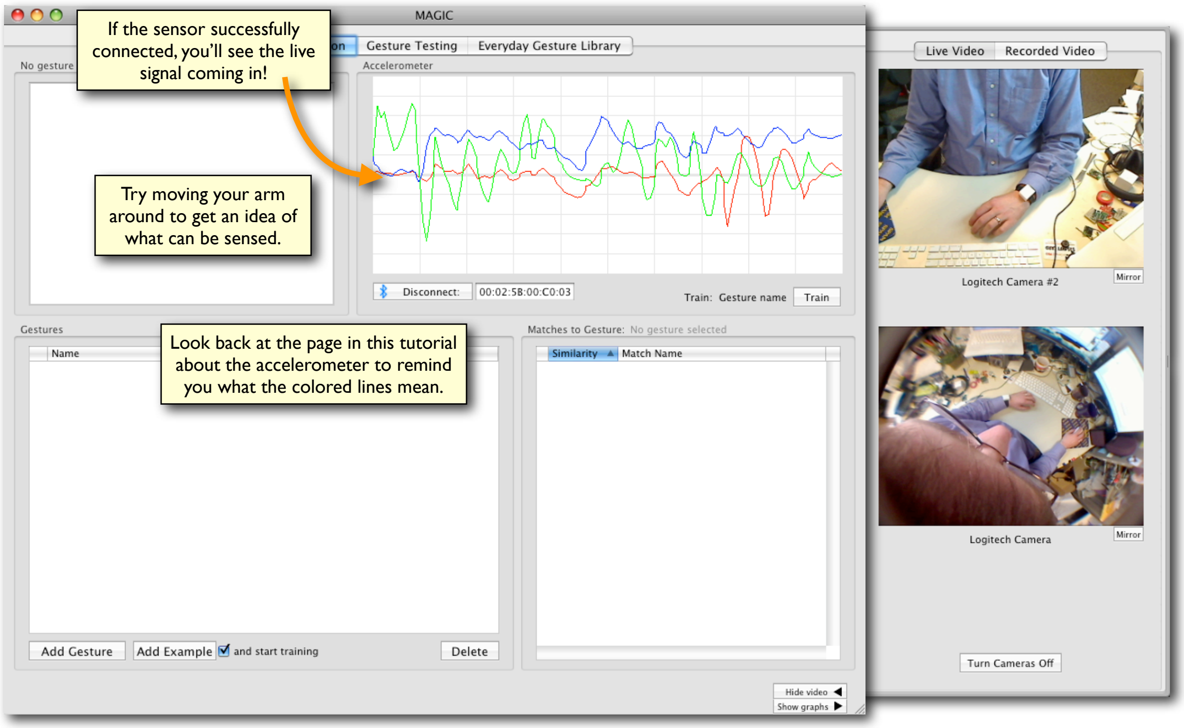
As you create gestures, the motions you make will be automatically recorded by the cameras. You'll be able to go back later and watch the video to help you remember the motions you were making.

This recorded video for each gesture will show up in the "Recorded Video" tab on the Video panel.

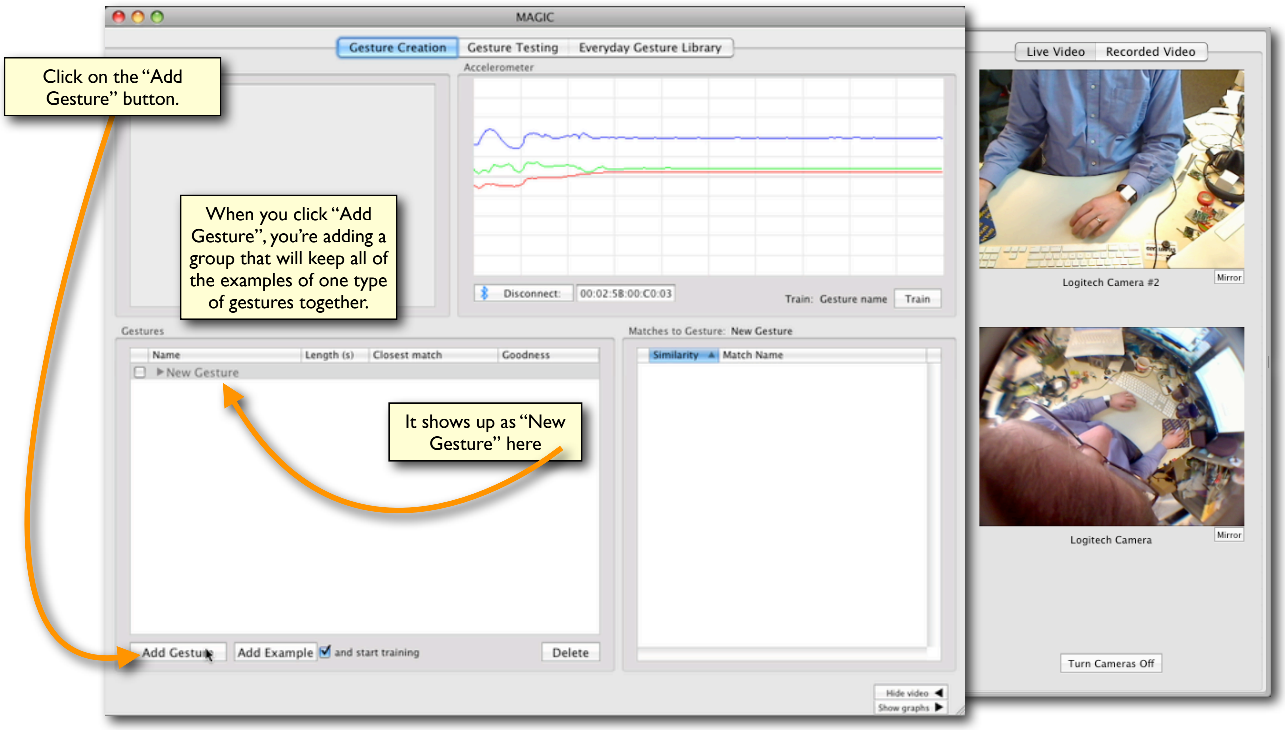
Let's make your first gesture!

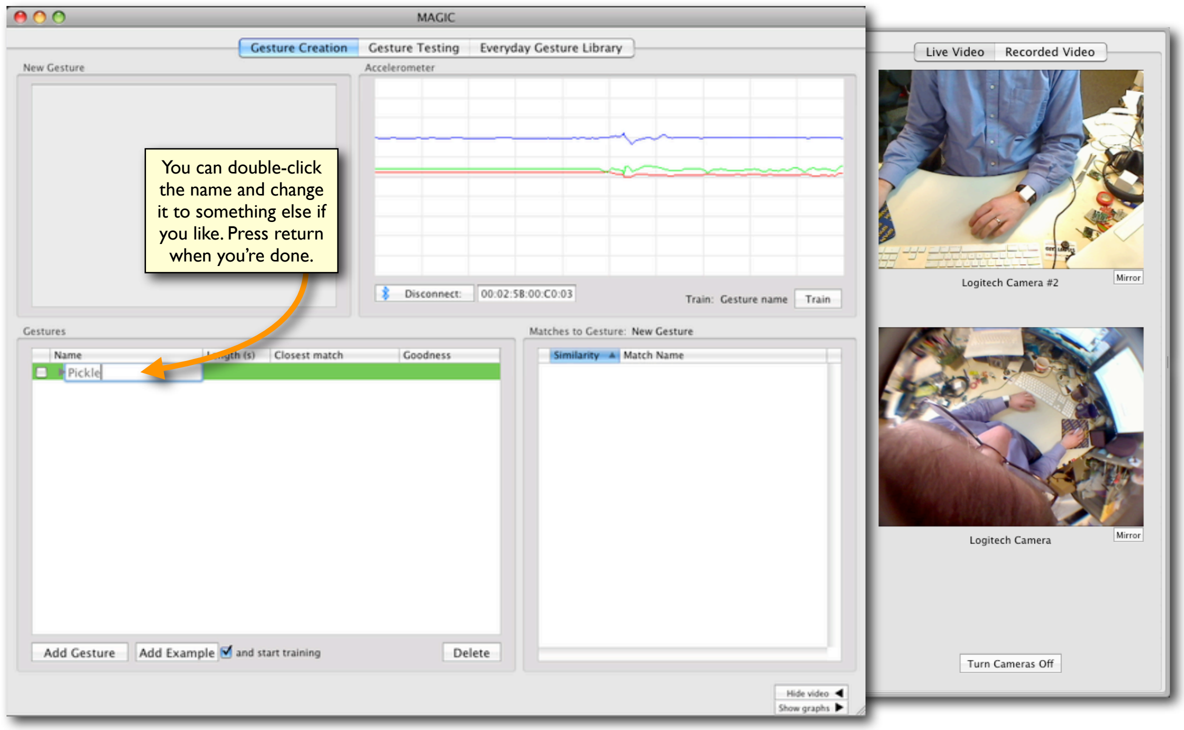


To begin, click on the *Connect to:* button, to let the computer talk to the wireless sensor.

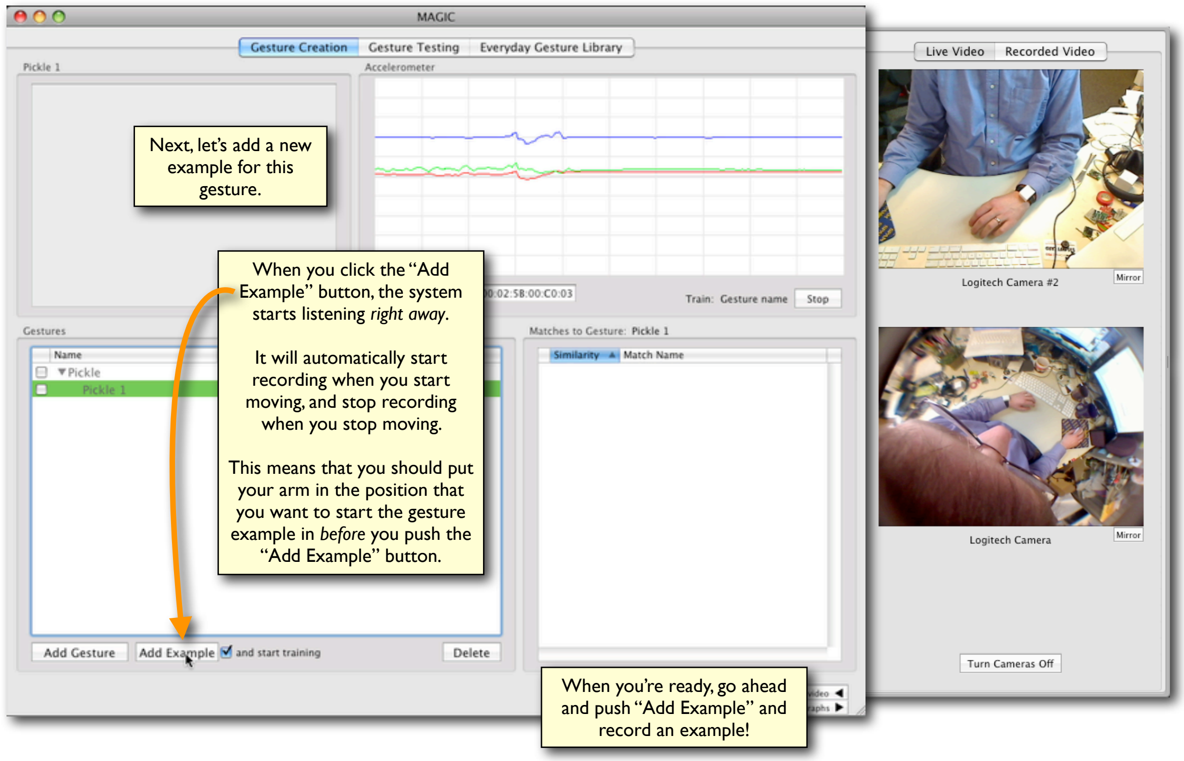


Let's try making a gesture!





You can double-click the name and change it to something else if you like. Press return when you're done.



Next, let's add a new example for this gesture.

When you click the "Add Example" button, the system starts listening *right away*.

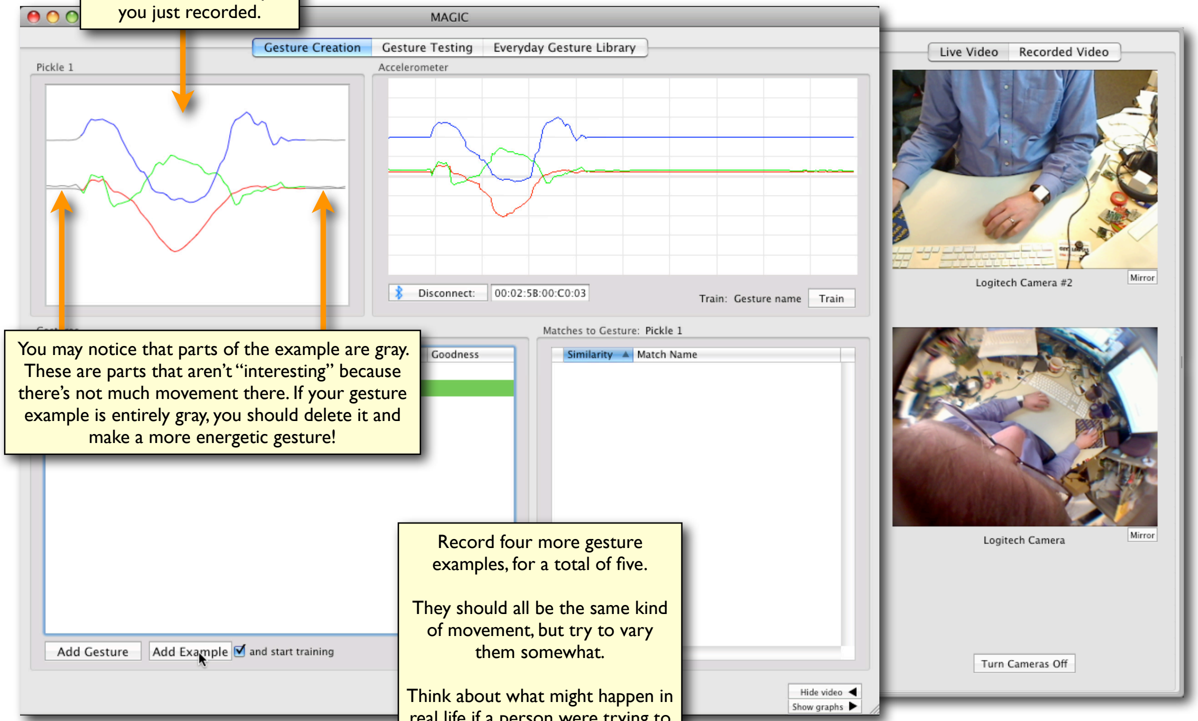
It will automatically start recording when you start moving, and stop recording when you stop moving.

This means that you should put your arm in the position that you want to start the gesture example in *before* you push the "Add Example" button.

When you're ready, go ahead and push "Add Example" and record an example!

After you stop moving your arm, you'll see something that looks like this screen.

This shows the example you just recorded.



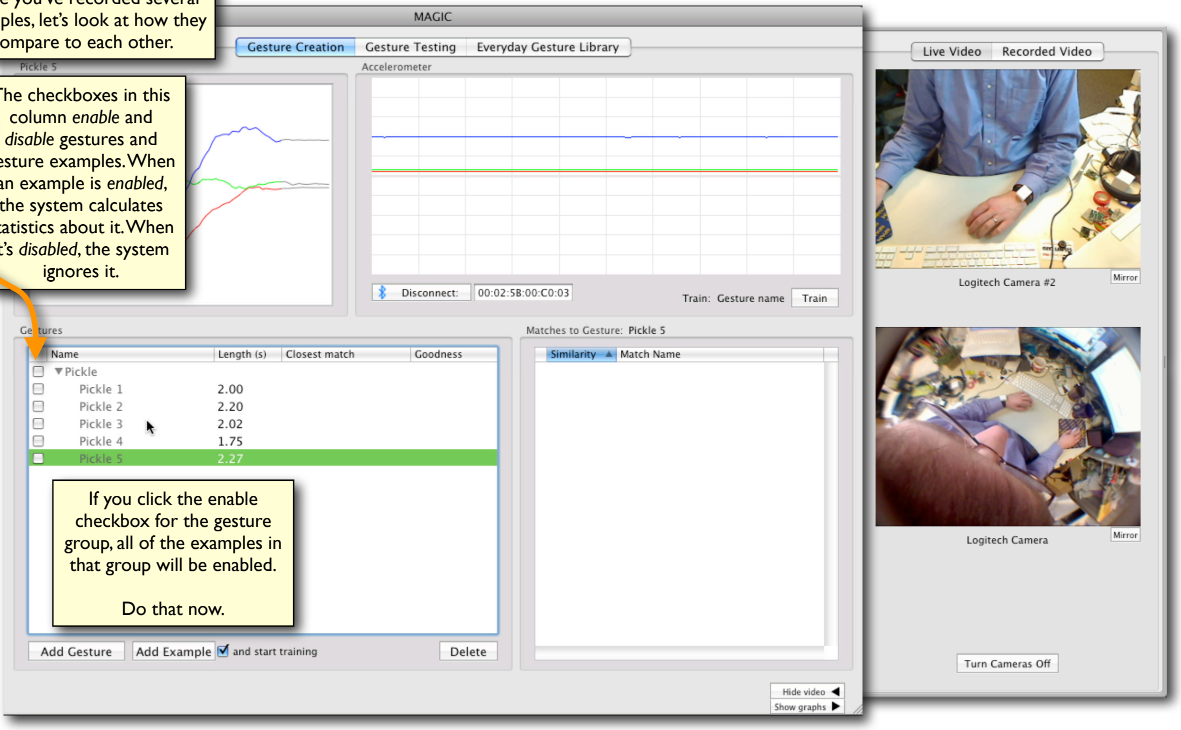
You may notice that parts of the example are gray. These are parts that aren't "interesting" because there's not much movement there. If your gesture example is entirely gray, you should delete it and make a more energetic gesture!

Record four more gesture examples, for a total of five.

They should all be the same kind of movement, but try to vary them somewhat.

Think about what might happen in real life if a person were trying to use your gesture—might they do it faster, or slower, or hold their arm slightly differently? Try to make examples that account for such possibilities.

Once you've recorded several examples, let's look at how they compare to each other.



	Name	Length (s)	Closest match	Goodness
☐	▼Pickle			
☐	Pickle 1	2.00		
☐	Pickle 2	2.20		
☐	Pickle 3	2.02		
☐	Pickle 4	1.75		
☐	Pickle 5	2.27		

The checkboxes in this column *enable* and *disable* gestures and gesture examples. When an example is *enabled*, the system calculates statistics about it. When it's *disabled*, the system ignores it.

If you click the enable checkbox for the gesture group, all of the examples in that group will be enabled.

Do that now.

Now you'll see some information in the columns next to each of your gesture examples.

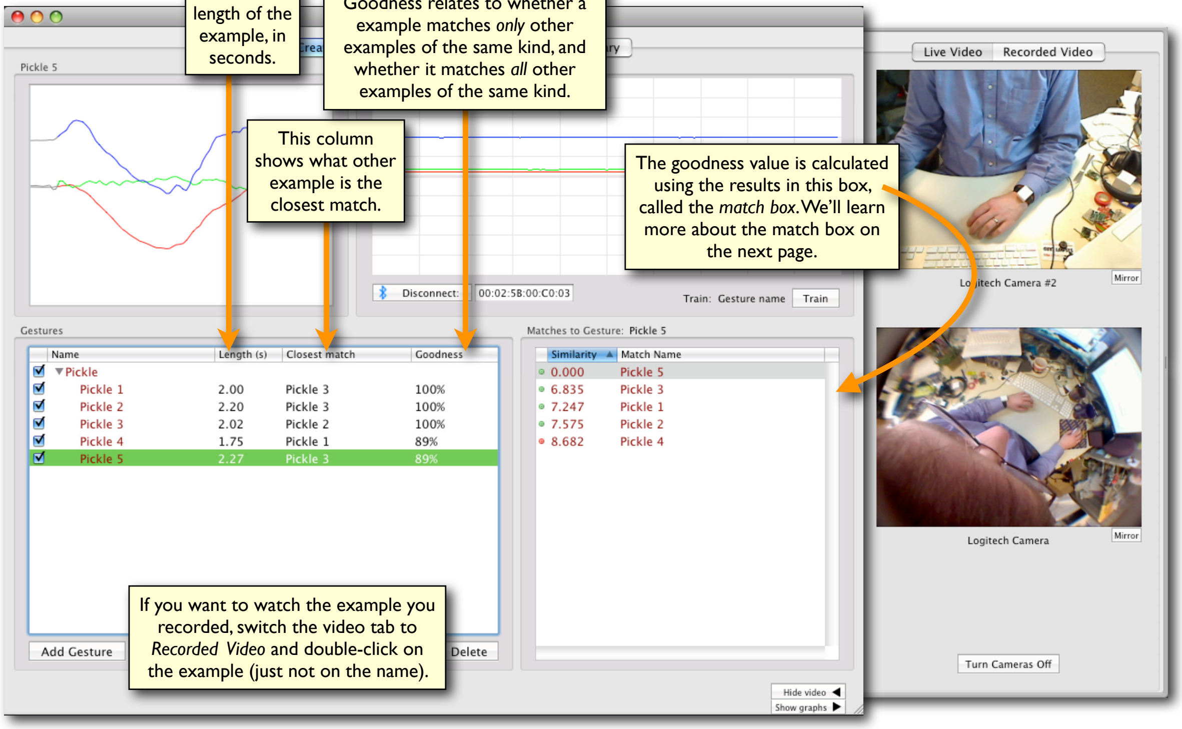
This column shows the length of the example, in seconds.

This column is a measure of the "goodness" of a gesture example. Goodness relates to whether a example matches *only* other examples of the same kind, and whether it matches *all* other examples of the same kind.

This column shows what other example is the closest match.

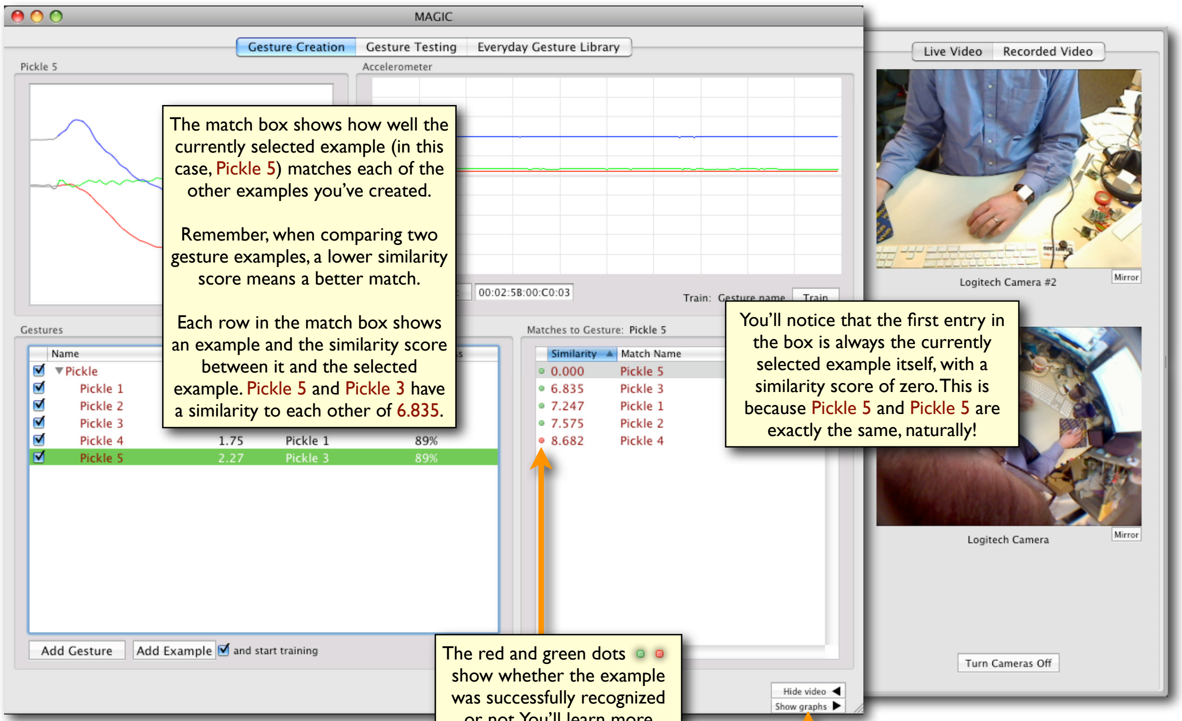
The goodness value is calculated using the results in this box, called the *match box*. We'll learn more about the match box on the next page.

If you want to watch the example you recorded, switch the video tab to *Recorded Video* and double-click on the example (just not on the name).



	Name	Length (s)	Closest match	Goodness
✓	▼Pickle			
✓	Pickle 1	2.00	Pickle 3	100%
✓	Pickle 2	2.20	Pickle 3	100%
✓	Pickle 3	2.02	Pickle 2	100%
✓	Pickle 4	1.75	Pickle 1	89%
✓	Pickle 5	2.27	Pickle 3	89%

Similarity ▲	Match Name
● 0.000	Pickle 5
● 6.835	Pickle 3
● 7.247	Pickle 1
● 7.575	Pickle 2
● 8.682	Pickle 4



	Name	Length (s)	Closest match	Goodness
✓	▼Pickle			
✓	Pickle 1			
✓	Pickle 2			
✓	Pickle 3			
✓	Pickle 4	1.75	Pickle 1	89%
✓	Pickle 5	2.27	Pickle 3	89%

Similarity ▲	Match Name
● 0.000	Pickle 5
● 6.835	Pickle 3
● 7.247	Pickle 1
● 7.575	Pickle 2
● 8.682	Pickle 4

The match box shows how well the currently selected example (in this case, Pickle 5) matches each of the other examples you've created.

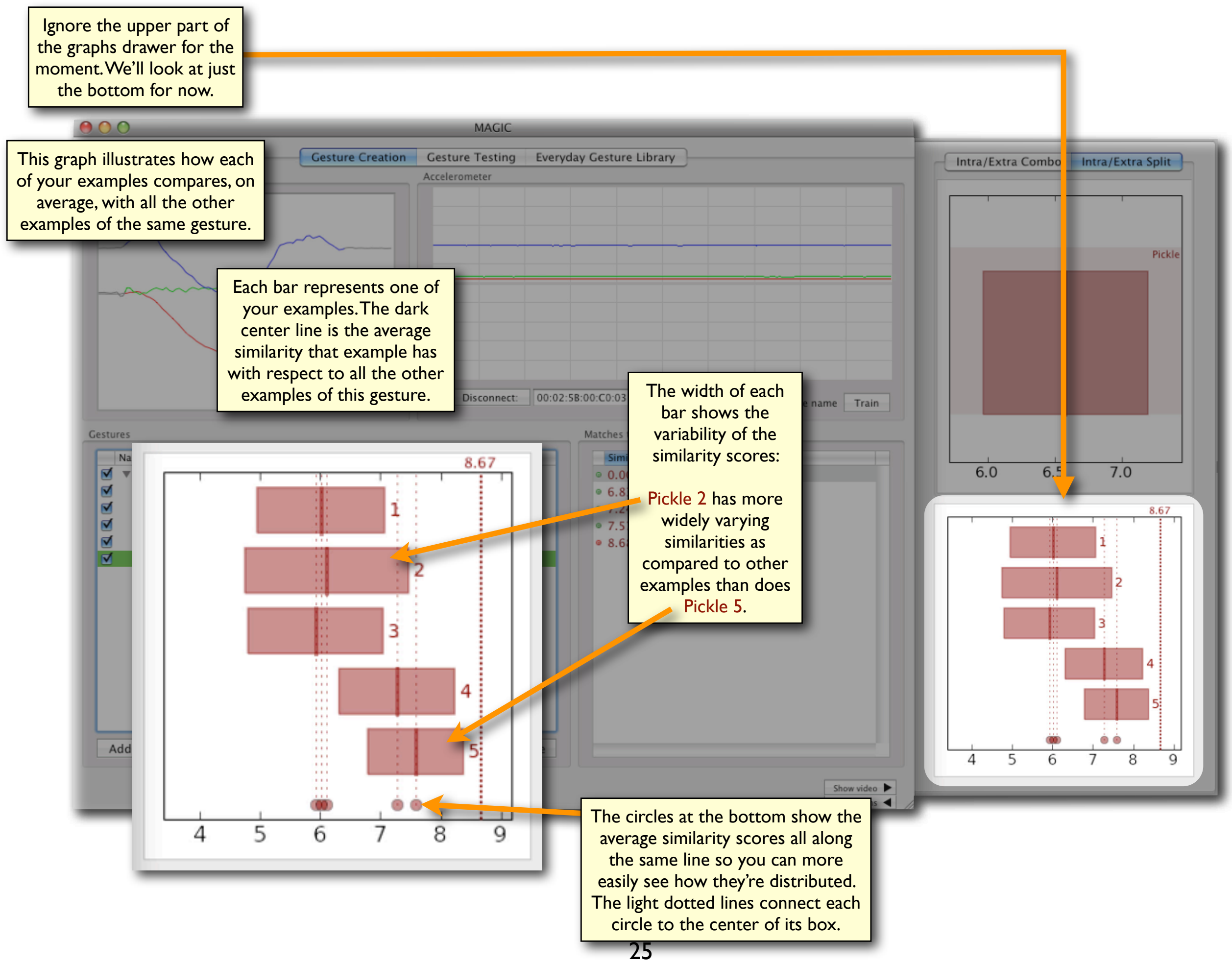
Remember, when comparing two gesture examples, a lower similarity score means a better match.

Each row in the match box shows an example and the similarity score between it and the selected example. Pickle 5 and Pickle 3 have a similarity to each other of 6.835.

You'll notice that the first entry in the box is always the currently selected example itself, with a similarity score of zero. This is because Pickle 5 and Pickle 5 are exactly the same, naturally!

The red and green dots ● ● show whether the example was successfully recognized or not. You'll learn more about this is determined on the next page.

First, click the "Show graphs" button.



Ignore the upper part of the graphs drawer for the moment. We'll look at just the bottom for now.

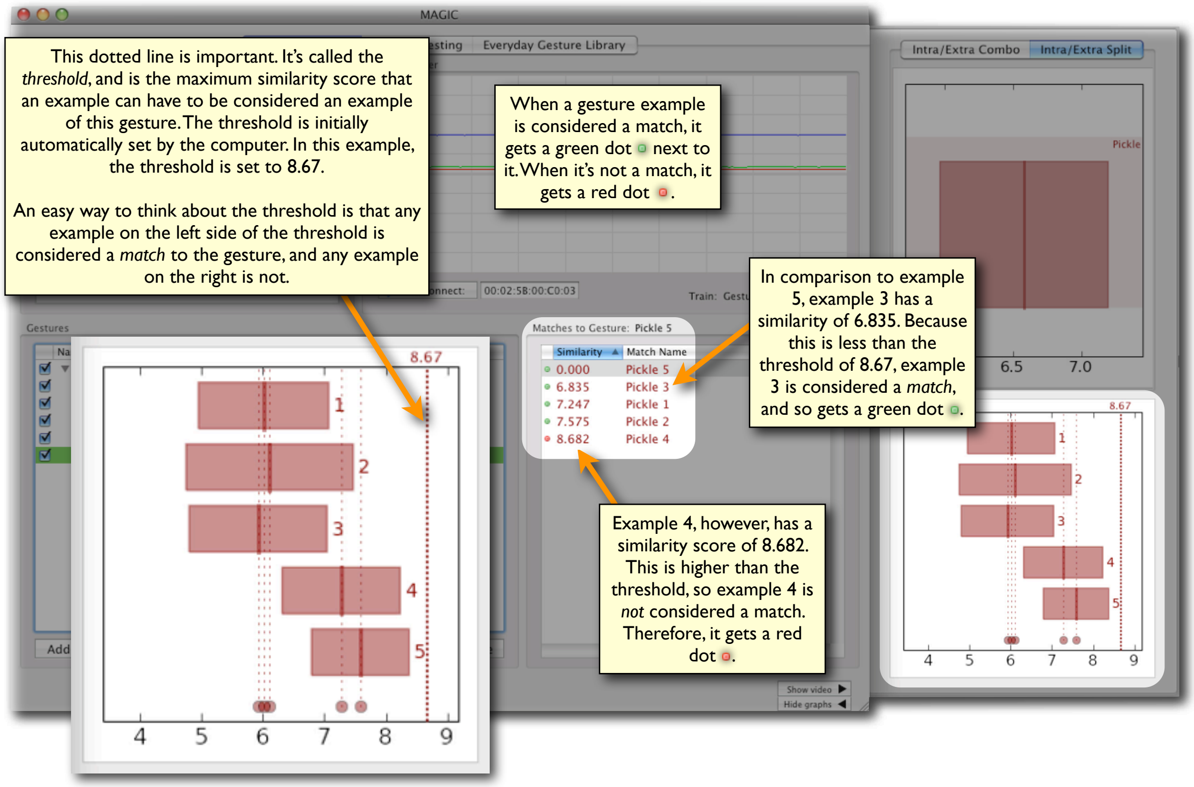
This graph illustrates how each of your examples compares, on average, with all the other examples of the same gesture.

Each bar represents one of your examples. The dark center line is the average similarity that example has with respect to all the other examples of this gesture.

The width of each bar shows the variability of the similarity scores:

Pickle 2 has more widely varying similarities as compared to other examples than does Pickle 5.

The circles at the bottom show the average similarity scores all along the same line so you can more easily see how they're distributed. The light dotted lines connect each circle to the center of its box.



Similarity	Match Name
0.000	Pickle 5
6.835	Pickle 3
7.247	Pickle 1
7.575	Pickle 2
8.682	Pickle 4

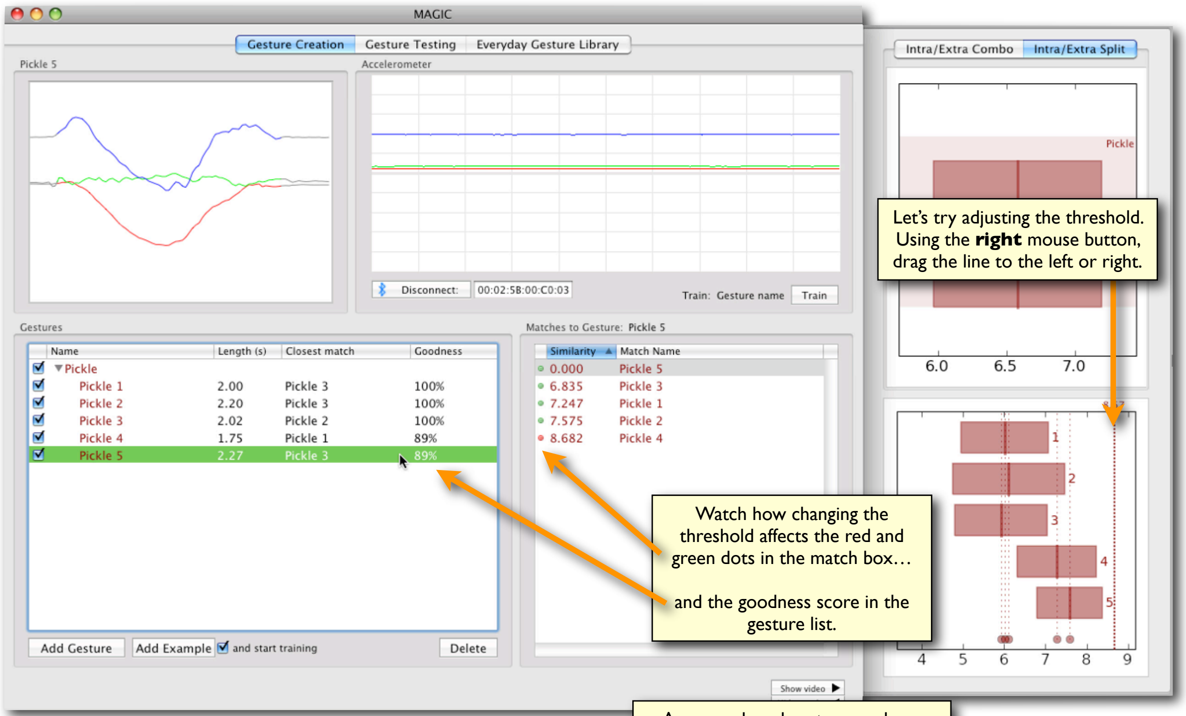
This dotted line is important. It's called the *threshold*, and is the maximum similarity score that an example can have to be considered an example of this gesture. The threshold is initially automatically set by the computer. In this example, the threshold is set to 8.67.

An easy way to think about the threshold is that any example on the left side of the threshold is considered a *match* to the gesture, and any example on the right is not.

When a gesture example is considered a match, it gets a green dot ● next to it. When it's not a match, it gets a red dot ●.

In comparison to example 5, example 3 has a similarity of 6.835. Because this is less than the threshold of 8.67, example 3 is considered a *match*, and so gets a green dot ●.

Example 4, however, has a similarity score of 8.682. This is higher than the threshold, so example 4 is *not* considered a match. Therefore, it gets a red dot ●.



Gestures

Name	Length (s)	Closest match	Goodness
▼Pickle			
Pickle 1	2.00	Pickle 3	100%
Pickle 2	2.20	Pickle 3	100%
Pickle 3	2.02	Pickle 2	100%
Pickle 4	1.75	Pickle 1	89%
Pickle 5	2.27	Pickle 3	89%

Matches to Gesture: Pickle 5

Similarity ▲	Match Name
● 0.000	Pickle 5
● 6.835	Pickle 3
● 7.247	Pickle 1
● 7.575	Pickle 2
● 8.682	Pickle 4

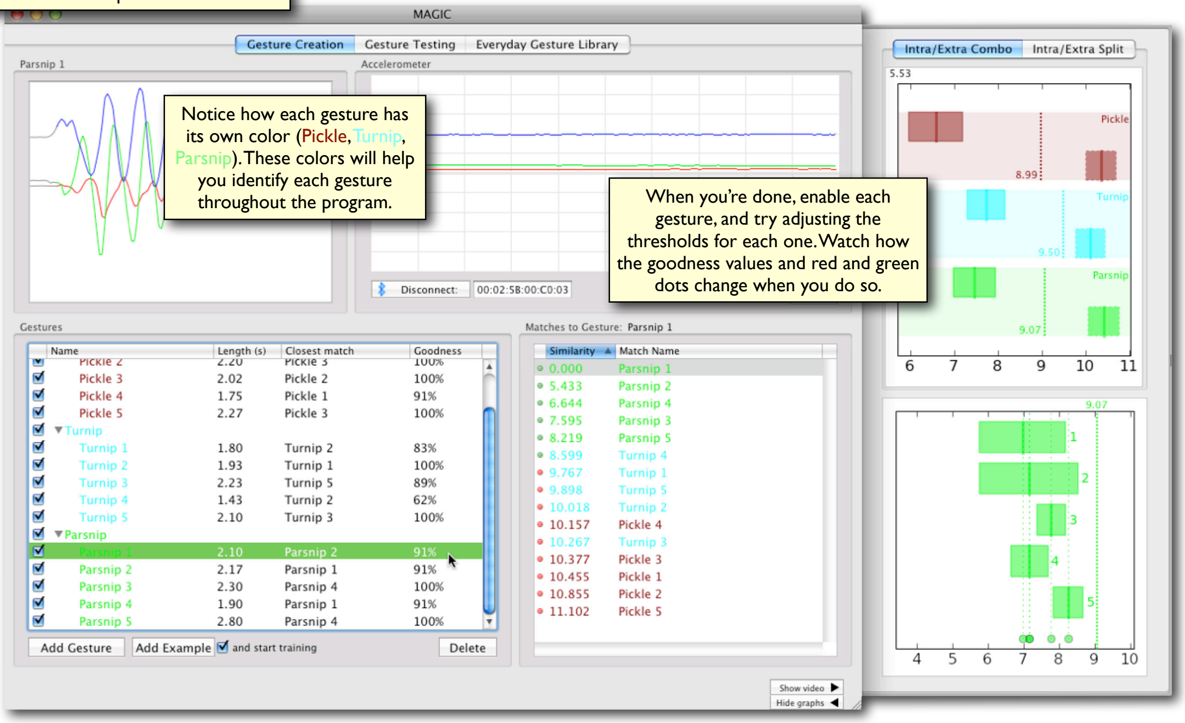
Let's try adjusting the threshold. Using the **right** mouse button, drag the line to the left or right.

Watch how changing the threshold affects the red and green dots in the match box…

and the goodness score in the gesture list.

An example only gets a goodness score of 100% if it matches *all* other examples in its category, and *no other* examples from other categories.

Now add two more gestures. Make five examples for each.



Notice how each gesture has its own color (Pickle, Turnip, Parsnip). These colors will help you identify each gesture throughout the program.

When you're done, enable each gesture, and try adjusting the thresholds for each one. Watch how the goodness values and red and green dots change when you do so.

Gestures

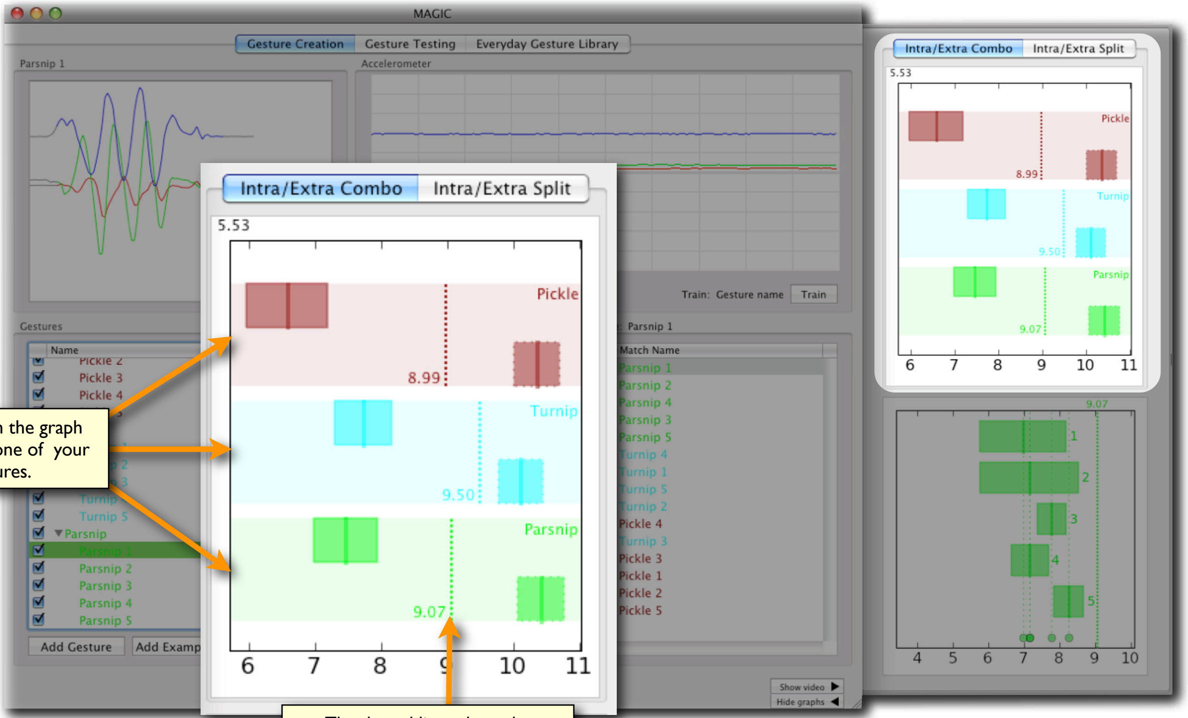
Name	Length (s)	Closest match	Goodness
Pickle 2	2.20	Pickle 3	100%
Pickle 3	2.02	Pickle 2	100%
Pickle 4	1.75	Pickle 1	91%
Pickle 5	2.27	Pickle 3	100%
▼Turnip			
Turnip 1	1.80	Turnip 2	83%
Turnip 2	1.93	Turnip 1	100%
Turnip 3	2.23	Turnip 5	89%
Turnip 4	1.43	Turnip 2	62%
Turnip 5	2.10	Turnip 3	100%
▼Parsnip			
Parsnip 1	2.10	Parsnip 2	91%
Parsnip 2	2.17	Parsnip 1	91%
Parsnip 3	2.30	Parsnip 4	100%
Parsnip 4	1.90	Parsnip 1	91%
Parsnip 5	2.80	Parsnip 4	100%

Matches to Gesture: Parsnip 1

Similarity ▲	Match Name
● 0.000	Parsnip 1
● 5.433	Parsnip 2
● 6.644	Parsnip 4
● 7.595	Parsnip 3
● 8.219	Parsnip 5
● 8.599	Turnip 4
● 9.767	Turnip 1
● 9.898	Turnip 5
● 10.018	Turnip 2
● 10.157	Pickle 4
● 10.267	Turnip 3
● 10.377	Pickle 3
● 10.455	Pickle 1
● 10.855	Pickle 2
● 11.102	Pickle 5

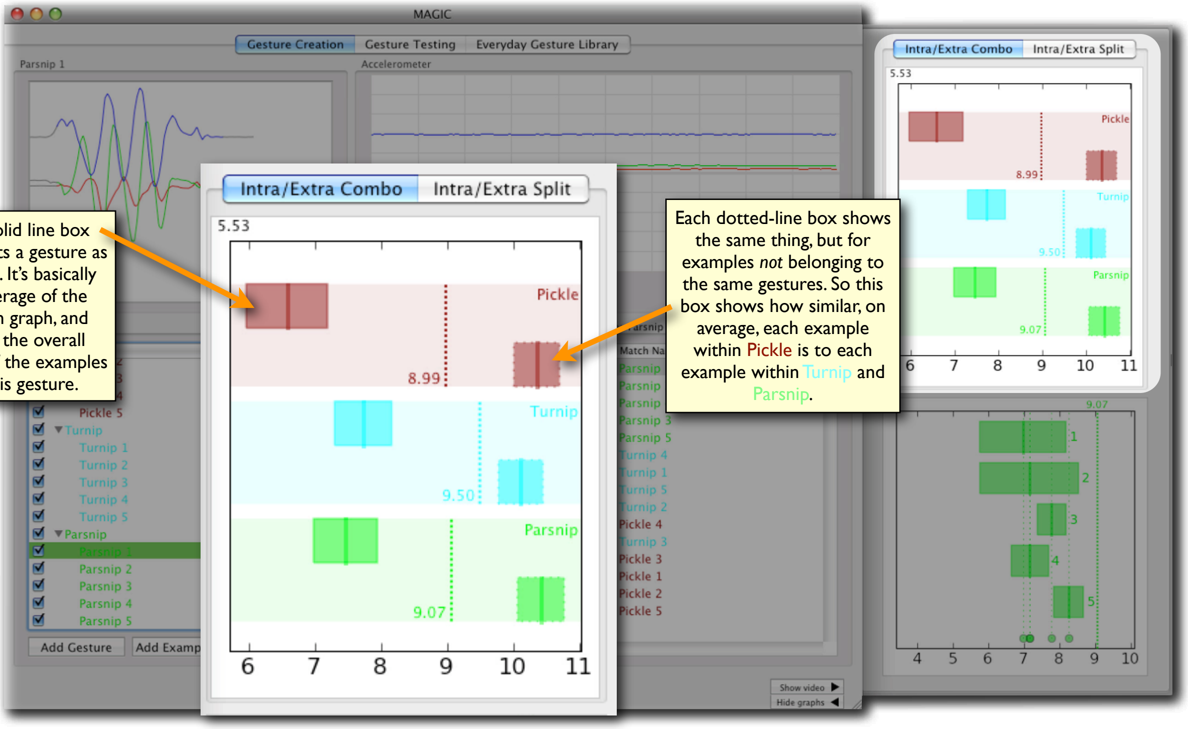
Now let's look at the top graph. Its purpose is to help you see how well each gesture as a whole can be distinguished from the other gestures you've made.

The top graph has two tabs. We'll first look at "Intra/Extra Combo."



Each row in the graph represents one of your gestures.

The dotted lines show the threshold for each gesture. You can adjust the thresholds here as well, using the **right** mouse button.

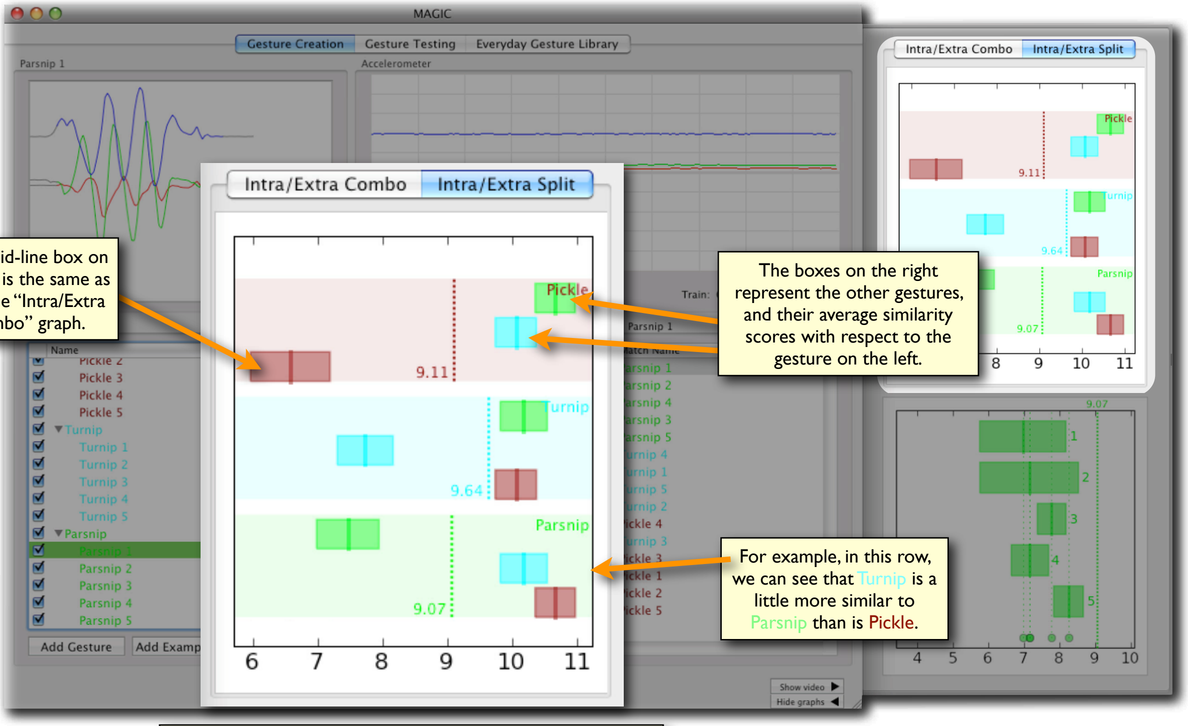


Each solid line box represents a gesture as a whole. It's basically the average of the bottom graph, and shows the overall spread of the examples for this gesture.

Each dotted-line box shows the same thing, but for examples *not* belonging to the same gestures. So this box shows how similar, on average, each example within Pickle is to each example within Turnip and Parsnip.

The point of this graph is to help you figure out whether your gestures as a whole are too similar to each other. If, for example, Pickle's two boxes were very close together, that might indicate that Pickle and other gestures could be confused with each other.

Next, let's look at "Intra/Extra Split."



The solid-line box on the left is the same as with the "Intra/Extra Combo" graph.

The boxes on the right represent the other gestures, and their average similarity scores with respect to the gesture on the left.

For example, in this row, we can see that Turnip is a little more similar to Parsnip than is Pickle.

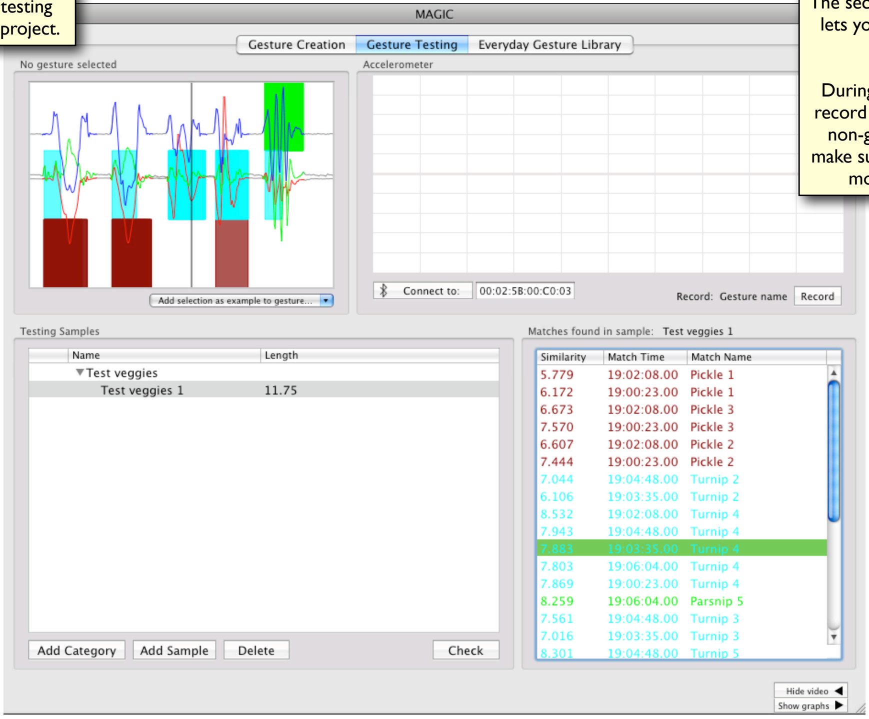
The idea behind this graph is to help you figure out if one of your gesture is particularly confusable with another one. If you need to, you can adjust the threshold line here as well. For example, the threshold for Turnip might be a little too close to the boxes for Pickle and Parsnip, so it might be a good idea to drag it to the left a bit.

31

Introduction to MAGIC

The Gesture Testing tab

This image shows the testing screen of MAGIC, mid-project.



Connect to: 00:02:5B:00:C0:03

Record: Gesture name

Testing Samples

Name	Length
▼Test veggies	
Test veggies 1	11.75

Matches found in sample: Test veggies 1

Similarity	Match Time	Match Name
5.779	19:02:08.00	Pickle 1
6.172	19:00:23.00	Pickle 1
6.673	19:02:08.00	Pickle 3
7.570	19:00:23.00	Pickle 3
6.607	19:02:08.00	Pickle 2
7.444	19:00:23.00	Pickle 2
7.044	19:04:48.00	Turnip 2
6.106	19:03:35.00	Turnip 2
8.532	19:02:08.00	Turnip 4
7.943	19:04:48.00	Turnip 4
7.883	19:03:35.00	Turnip 4
7.803	19:06:04.00	Turnip 4
7.869	19:00:23.00	Turnip 4
8.259	19:06:04.00	Parsnip 5
7.561	19:04:48.00	Turnip 3
7.016	19:03:35.00	Turnip 3
8.301	19:04:48.00	Turnip 5

The second tab, Gesture Testing, lets you test out the gestures you've created.

During testing, you'll typically record a variety of gesture and non-gesture movements, to make sure that only the desired motions are detected.

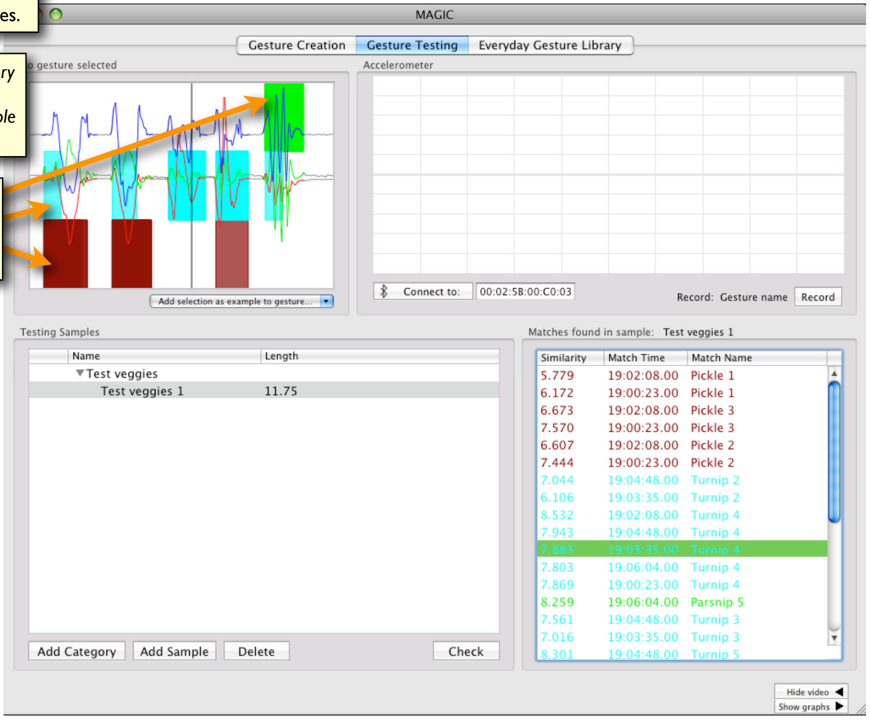
The next few pages in this tutorial will introduce the parts of the Gesture Testing tab.

The purpose of the testing tab is to make sure that the gestures you've created are properly recognized. It lets you test out your gestures without necessarily creating new ones.

In this picture, a test *category* ("Test veggies") has been created, with one test *sample* ("Test veggies 1").

The colored blocks show places in the test sample where gestures were found.

Next, you'll learn about the different parts of the testing tab.



Connect to: 00:02:5B:00:C0:03

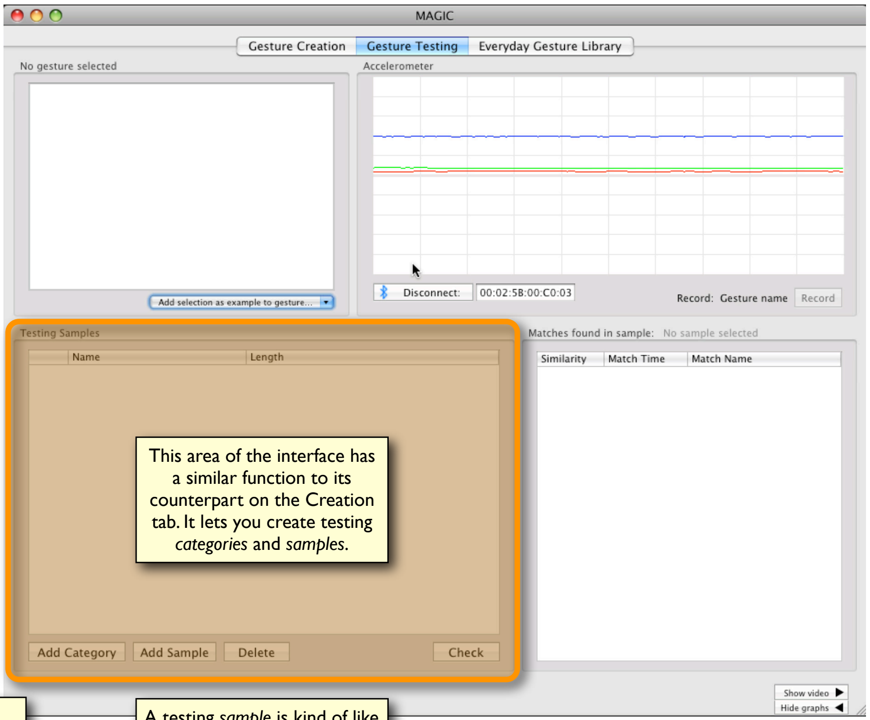
Record: Gesture name

Testing Samples

Name	Length
▼Test veggies	
Test veggies 1	11.75

Matches found in sample: Test veggies 1

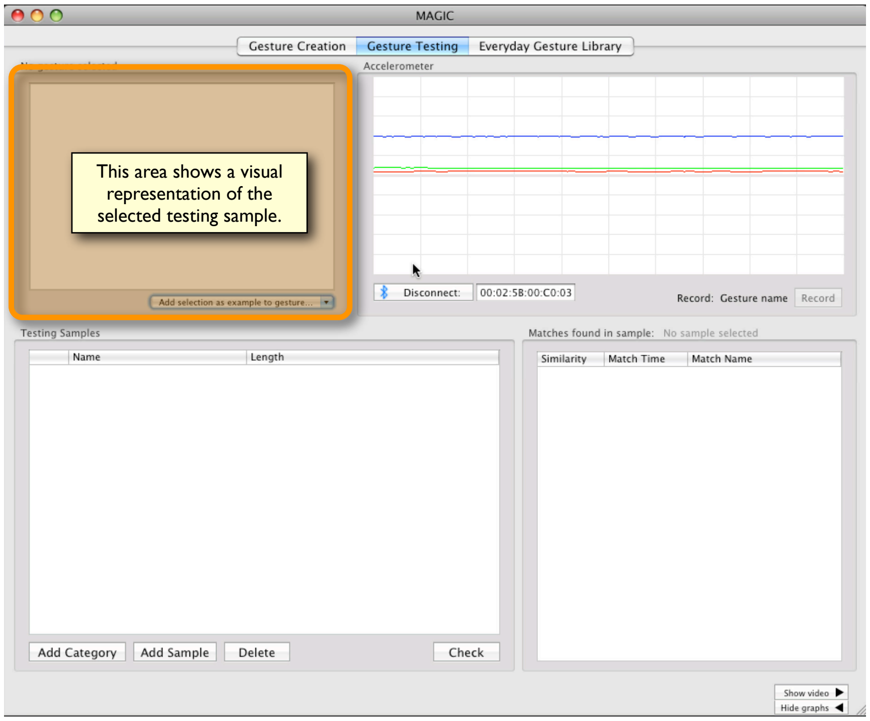
Similarity	Match Time	Match Name
5.779	19:02:08.00	Pickle 1
6.172	19:00:23.00	Pickle 1
6.673	19:02:08.00	Pickle 3
7.570	19:00:23.00	Pickle 3
6.607	19:02:08.00	Pickle 2
7.444	19:00:23.00	Pickle 2
7.044	19:04:48.00	Turnip 2
6.106	19:03:35.00	Turnip 2
8.532	19:02:08.00	Turnip 4
7.943	19:04:48.00	Turnip 4
7.883	19:03:35.00	Turnip 4
7.803	19:06:04.00	Turnip 4
7.869	19:00:23.00	Turnip 4
8.259	19:06:04.00	Parsnip 5
7.561	19:04:48.00	Turnip 3
7.016	19:03:35.00	Turnip 3
8.301	19:04:48.00	Turnip 5



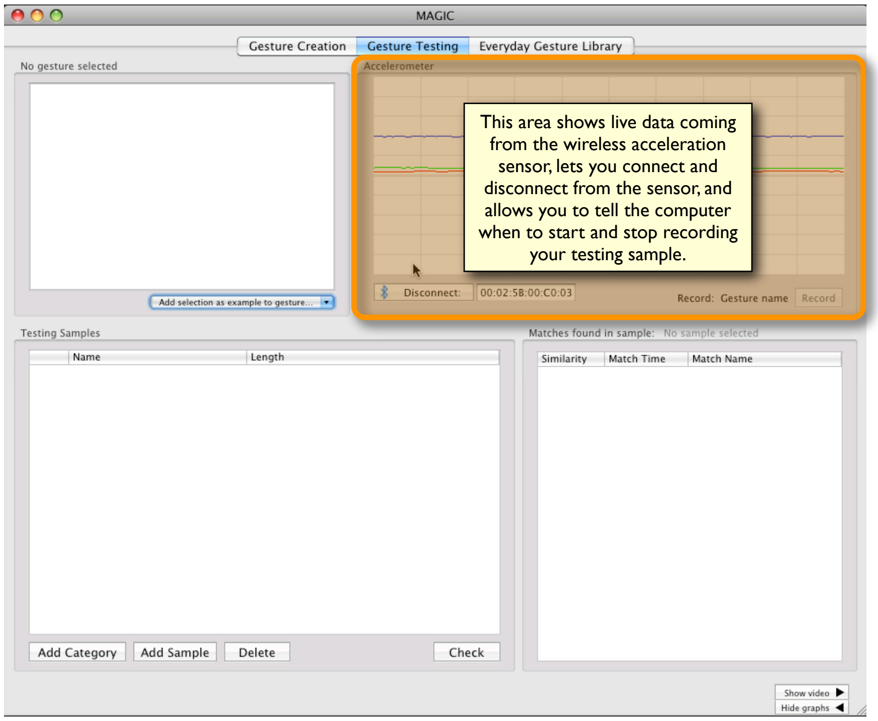
This area of the interface has a similar function to its counterpart on the Creation tab. It lets you create testing *categories* and *samples*.

A testing *category* is a container that holds similar test samples together.

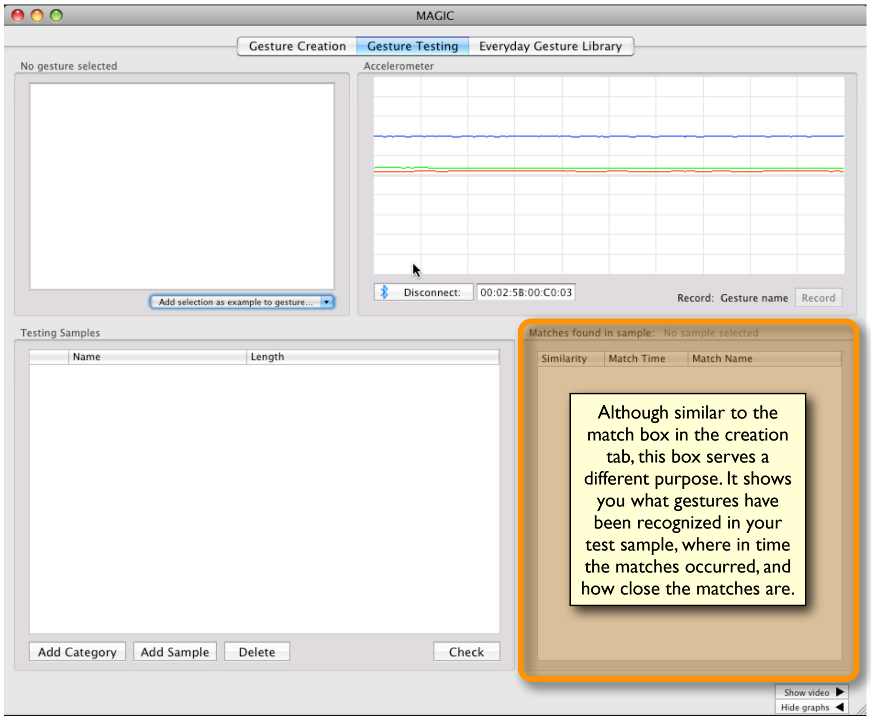
A testing *sample* is kind of like a gesture example. It's a recording of motion you make. Unlike a gesture example, however, a testing sample can contain multiple distinct movements.



This area shows a visual representation of the selected testing sample.



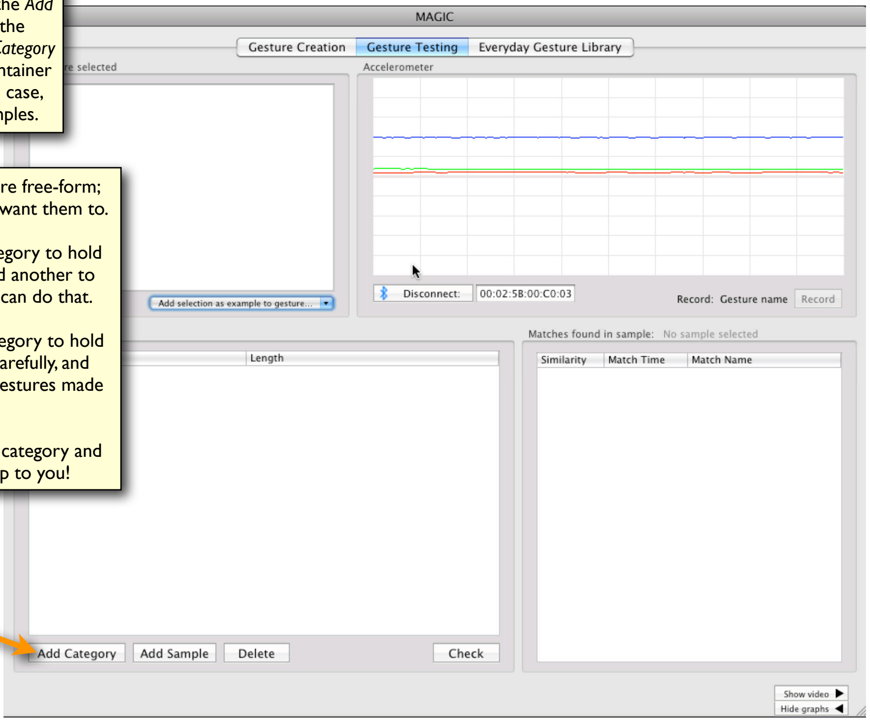
This area shows live data coming from the wireless acceleration sensor, lets you connect and disconnect from the sensor, and allows you to tell the computer when to start and stop recording your testing sample.



Although similar to the match box in the creation tab, this box serves a different purpose. It shows you what gestures have been recognized in your test sample, where in time the matches occurred, and how close the matches are.

Introduction to MAGIC

The gesture testing process



First, let's create a new testing category. Like the *Add Gesture* button on the creation tab, the *Add Category* button will make a container to hold things. In this case, you'll be holding samples.

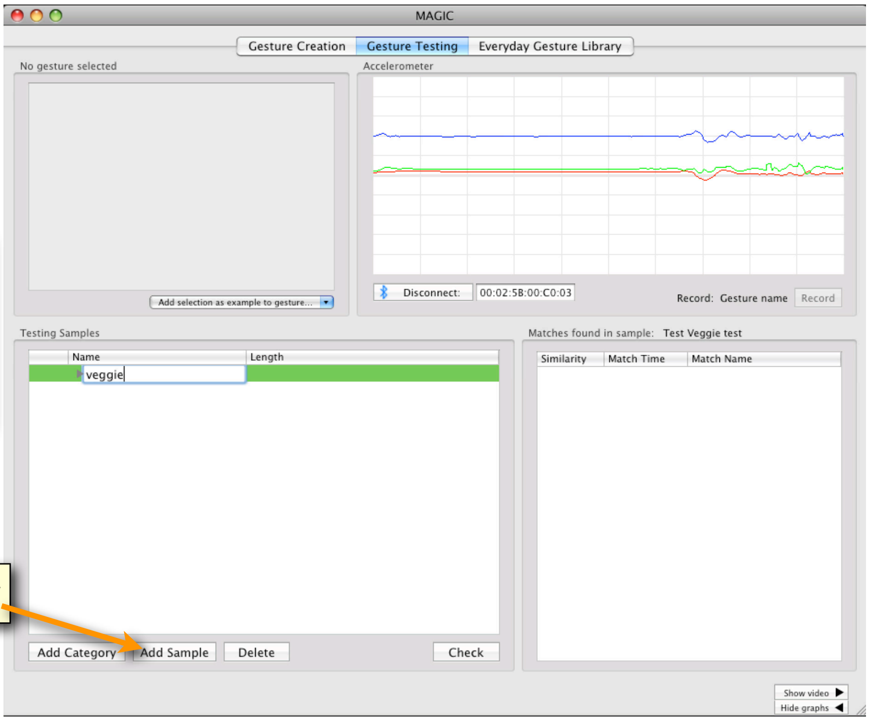
Unlike gestures, categories are free-form; they can mean whatever you want them to.

If you want to make one category to hold testing samples of Pickle and another to hold samples of Turnip, you can do that.

You could also make one category to hold samples of gestures done carefully, and another to hold samples of gestures made sloppily.

Or you could just make one category and put everything in it. It's up to you!

Click the *Add Category* button.



Name	Length
veggie	

Similarity	Match Time	Match Name

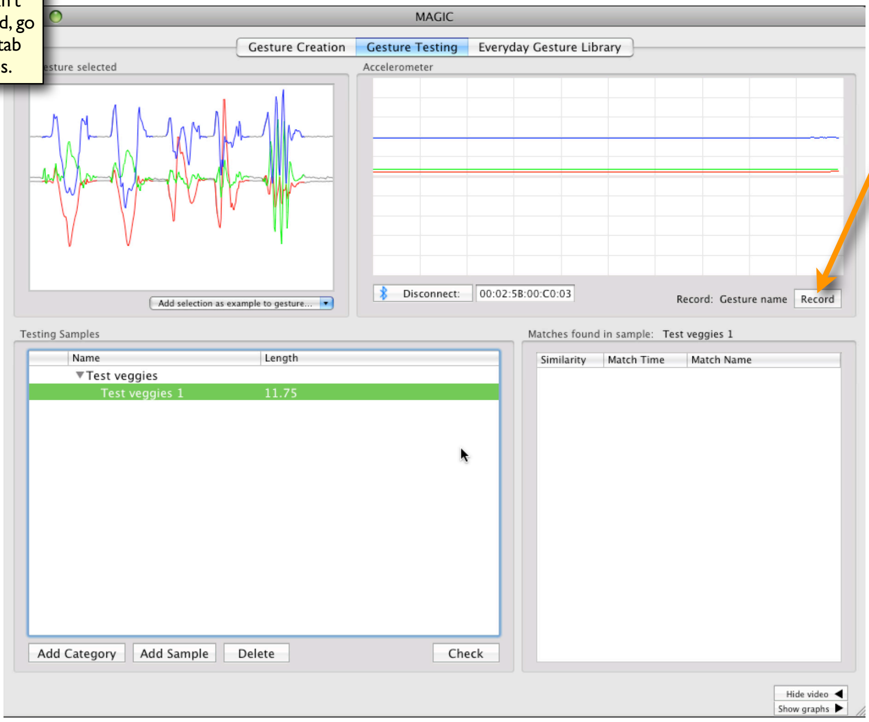
Like gestures on the creation tab, you can double-click the name of the category to rename it.

MAGIC will automatically add "Test" to the beginning of the name.

Now, let's add a new sample. Click the *Add Sample* button.

Unlike with the *Add Example* button, MAGIC will **not** automatically start recording.

Now, you'll record yourself performing the gestures you created on the creation tab. If you can't remember what you did, go back to the creation tab and watch the videos.



Name	Length
▼Test veggies	
Test veggies 1	11.75

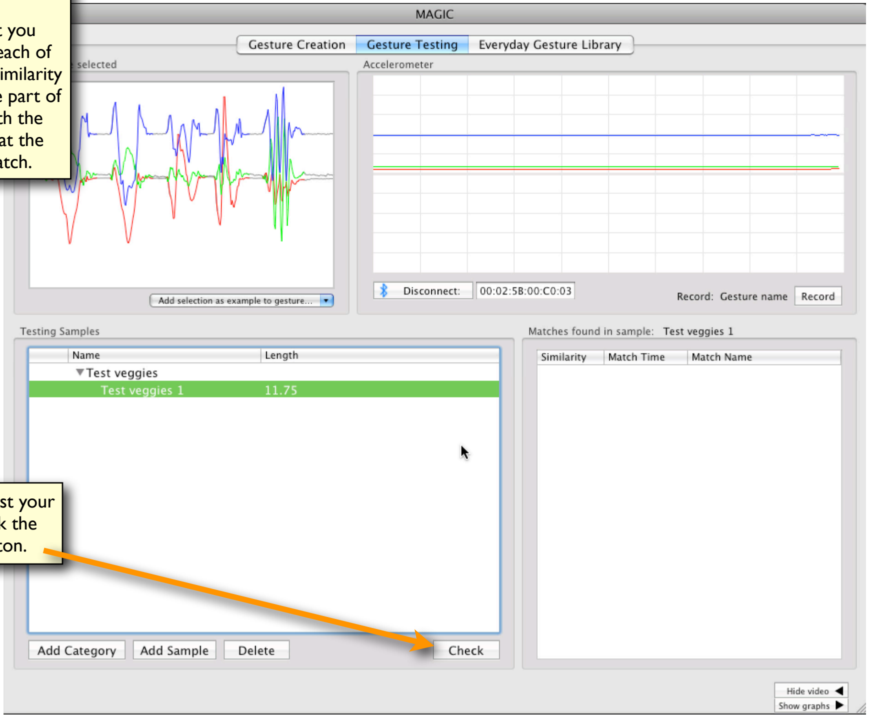
Similarity	Match Time	Match Name

When you're ready, click the *Record* button. The system will start recording your gestures, and the *Record* button will change to the *Stop* button. Click it to stop recording when you're finished.

The next step is to determine if your gestures were recognized.

The recognition process works similarly to how gesture examples are compared on the creation tab.

Each part of the movement you recorded is checked against each of your gesture examples. If the similarity score between an example the part of the test sample is underneath the threshold for the gesture that the example is part of, it's a match.



Name	Length
▼Test veggies	
Test veggies 1	11.75

Similarity	Match Time	Match Name

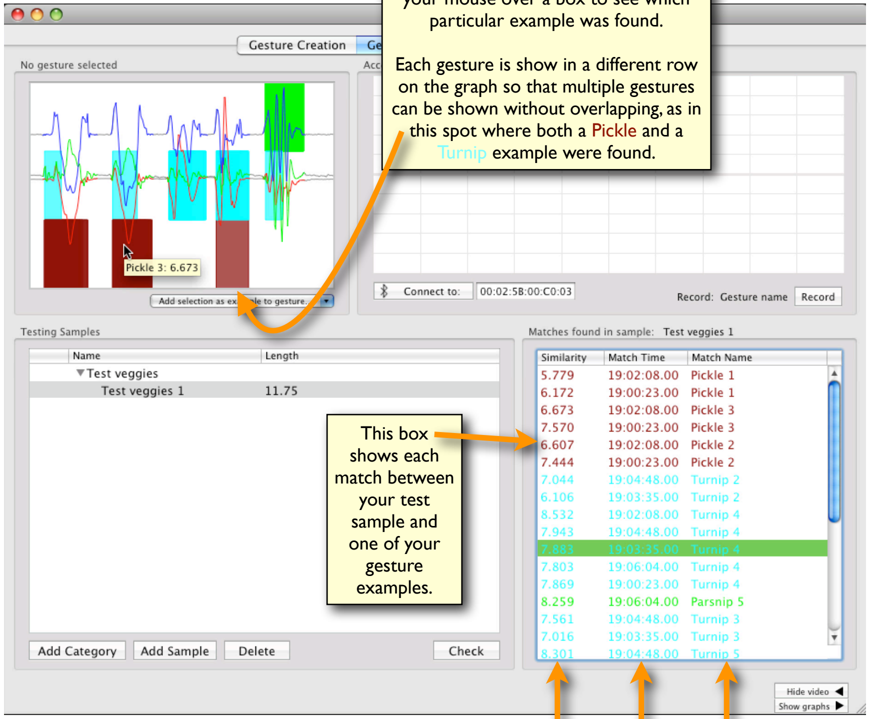
In order to test your sample, click the *Check* button.

43

When matches between your gesture examples and your test sample are found, they're highlighted with colored boxes.

The color of the box corresponds to the gesture found. You can also hover your mouse over a box to see which particular example was found.

Each gesture is show in a different row on the graph so that multiple gestures can be shown without overlapping, as in this spot where both a Pickle and a Turnip example were found.



Name	Length
▼Test veggies	
Test veggies 1	11.75

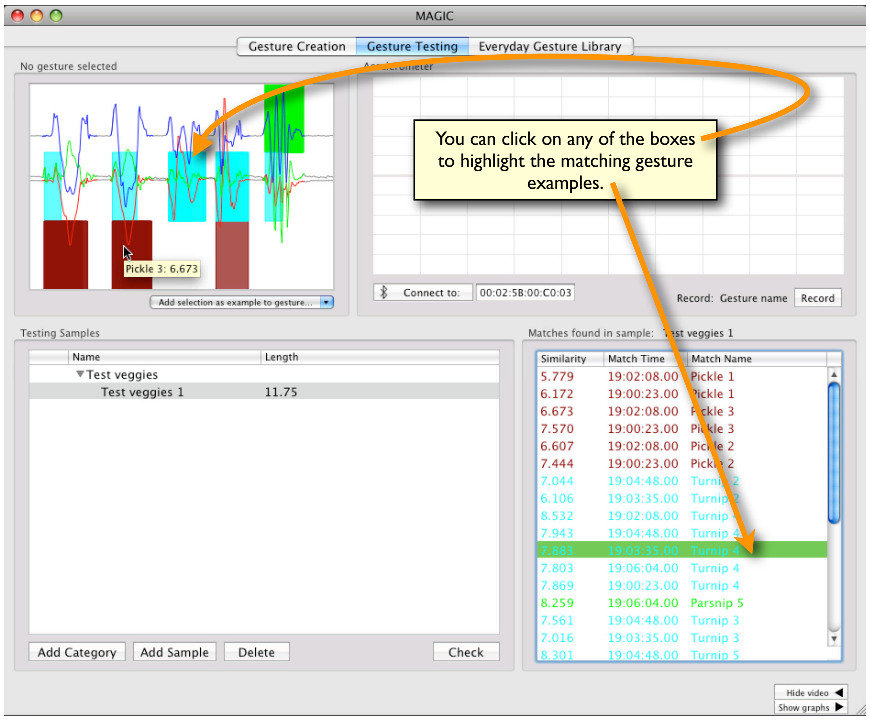
Similarity	Match Time	Match Name
5.779	19:02:08.00	Pickle 1
6.172	19:00:23.00	Pickle 1
6.673	19:02:08.00	Pickle 3
7.570	19:00:23.00	Pickle 3
6.607	19:02:08.00	Pickle 2
7.444	19:00:23.00	Pickle 2
7.044	19:04:48.00	Turnip 2
6.106	19:03:35.00	Turnip 2
8.532	19:02:08.00	Turnip 4
7.943	19:04:48.00	Turnip 4
7.883	19:03:35.00	Turnip 4
7.803	19:06:04.00	Turnip 4
7.869	19:00:23.00	Turnip 4
8.259	19:06:04.00	Parsnip 5
7.561	19:04:48.00	Turnip 3
7.016	19:03:35.00	Turnip 3
8.301	19:04:48.00	Turnip 5

This box shows each match between your test sample and one of your gesture examples.

It shows the similarity between the example and the part of your sample…

…the time within your sample at which the match was found…

…and the name of the example that matched the sample.



Name	Length
▼Test veggies	
Test veggies 1	11.75

Matches found in sample: Test veggies 1

Similarity	Match Time	Match Name
5.779	19:02:08.00	Pickle 1
6.172	19:00:23.00	Pickle 1
6.673	19:02:08.00	Pickle 3
7.570	19:00:23.00	Pickle 3
6.607	19:02:08.00	Pickle 2
7.444	19:00:23.00	Pickle 2
7.044	19:04:48.00	Turnip 2
6.106	19:03:35.00	Turnip 2
8.532	19:02:08.00	Turnip 4
7.943	19:04:48.00	Turnip 4
7.883	19:03:35.00	Turnip 4
7.803	19:06:04.00	Turnip 4
7.869	19:00:23.00	Turnip 4
8.259	19:06:04.00	Parsnip 5
7.561	19:04:48.00	Turnip 3
7.016	19:03:35.00	Turnip 3
8.301	19:04:48.00	Turnip 5

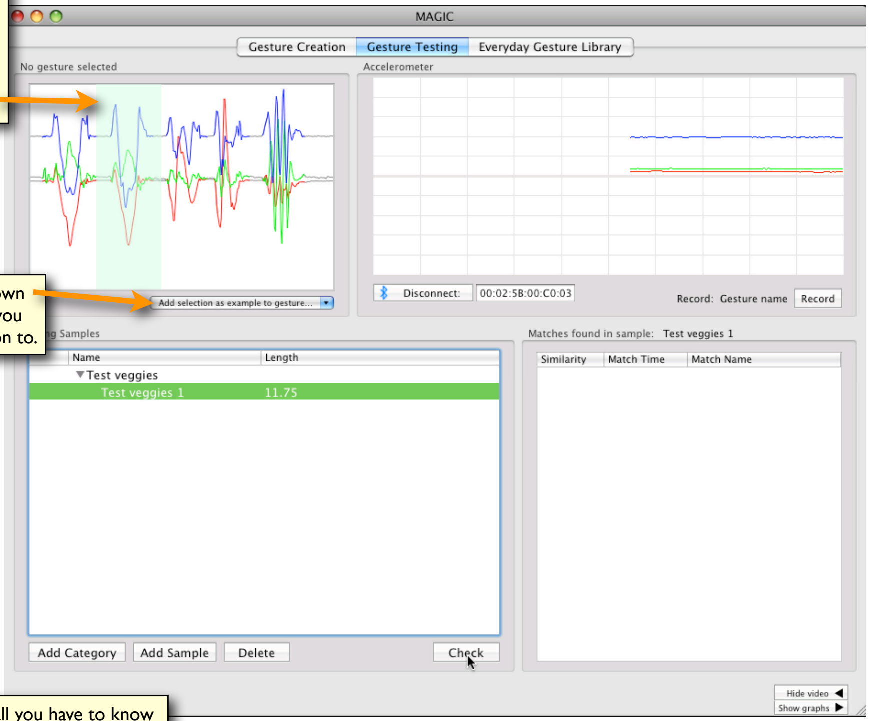
You can click on any of the boxes to highlight the matching gesture examples.

Or you can click a matching example, and the corresponding box will highlight.

If you have matches that are wrong, you can use the graph drawer to adjust the thresholds until you get the desired results. But be sure to check the creation tab to avoid undesired effects!

If you have a part of your test sample that *should* have been recognized, but wasn't, you can use it as a gesture example.

Using the **left** mouse button, drag to highlight the area that you want.



Name	Length
▼Test veggies	
Test veggies 1	11.75

Then, use this drop-down to select the gesture you want to add the selection to.

That's all you have to know about the testing tab!

Introduction to MAGIC

The Everyday Gesture Library

The Everyday Gesture Library (EGL)

One consideration when designing gestures is whether they'll be usable. An important aspect of usability is *everyday use*—that is, using the gestures while doing other things.

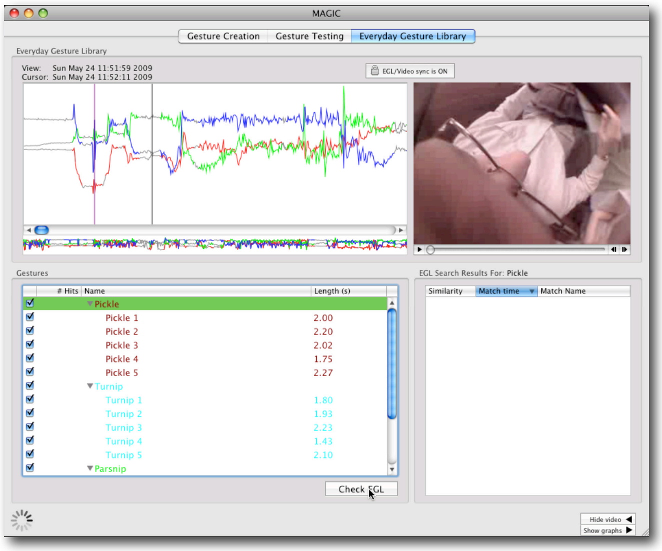
For our vegetable vending machine, we don't want customers' normal motions to accidentally dispense unwanted vegetables! This means that when Mr. Onara waves hello to Mrs. Ishuu, the machine shouldn't dump out a load of pickles!

How can you solve this problem? One way is to try to think of every possible movement that Mr. Onara might make, and avoid designing gestures like that. Then you can give the system to him—and other willing customers—to test, and see how many accidental vegetable deliveries you end up with.

The problem with this solution is that it's very time consuming (and wasteful of vegetables!). Every time you find that one of the gestures you made is too similar to a customer's usual movements, you have to design a new gesture and run the test all over again!

This is where the **E**veryday **G**esture **L**ibrary—or **EGL**—tab comes in. The EGL is a recording of a typical person's everyday movements. The person has simply worn the sensor and proceeded with his life, letting it record his motions. This person has also worn a hat with a camera, so you'll be able to see what kinds of motions were made.

The EGL tab works a lot like the testing tab, but you're provided with samples instead of making your own. You'll learn a bit more about the EGL tab on the next page.



View: Sun May 24 11:51:59 2009
Cursor: Sun May 24 11:52:11 2009

	# Hits	Name	Length (s)
✓		▼Pickle	
✓		Pickle 1	2.00
✓		Pickle 2	2.20
✓		Pickle 3	2.02
✓		Pickle 4	1.75
✓		Pickle 5	2.27
✓		▼Turnip	
✓		Turnip 1	1.80
✓		Turnip 2	1.93
✓		Turnip 3	2.23
✓		Turnip 4	1.43
✓		Turnip 5	2.10
✓		▼Parsnip	

Similarity	Match time	Match Name

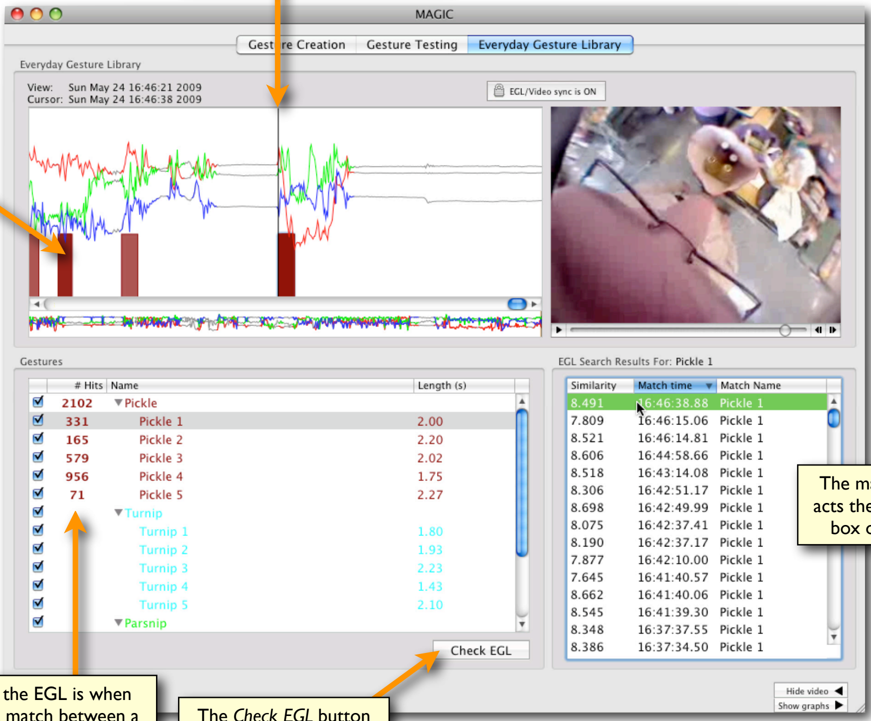
The EGL tab is laid out like a combination between the creation and testing tabs.

You can move this cursor through the EGL by dragging with your **right** mouse button. The video will follow along.

This part shows the stream of movements collected by the user, and, like the testing tab, shows boxes whenever a match is found. Also like the testing tab, you can click a box to select all of the matching results in the match box.

If you double-click an example, the video of you performing the gesture will play simultaneously with the EGL video.

The match box looks and acts the same as the match box on the testing tab.



View: Sun May 24 16:46:21 2009
Cursor: Sun May 24 16:46:38 2009

	# Hits	Name	Length (s)
✓	2102	▼Pickle	
✓	331	Pickle 1	2.00
✓	165	Pickle 2	2.20
✓	579	Pickle 3	2.02
✓	956	Pickle 4	1.75
✓	71	Pickle 5	2.27
✓		▼Turnip	
✓		Turnip 1	1.80
✓		Turnip 2	1.93
✓		Turnip 3	2.23
✓		Turnip 4	1.43
✓		Turnip 5	2.10
✓		▼Parsnip	

EGL Search Results For: Pickle 1

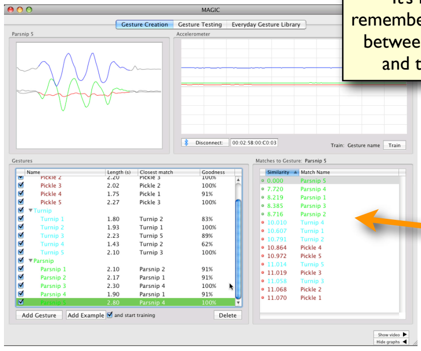
Similarity	Match time	Match Name
8.491	16:46:38.88	Pickle 1
7.809	16:46:15.06	Pickle 1
8.521	16:46:14.81	Pickle 1
8.606	16:44:58.66	Pickle 1
8.518	16:43:14.08	Pickle 1
8.306	16:42:51.17	Pickle 1
8.698	16:42:49.99	Pickle 1
8.075	16:42:37.41	Pickle 1
8.190	16:42:37.17	Pickle 1
7.877	16:42:10.00	Pickle 1
7.645	16:41:40.57	Pickle 1
8.662	16:41:40.06	Pickle 1
8.545	16:41:39.30	Pickle 1
8.348	16:37:37.55	Pickle 1
8.386	16:37:34.50	Pickle 1

A *hit* in the EGL is when there's a match between a gesture example and part of the EGL. This column shows you how many were found for each example, and for each gesture as a whole.

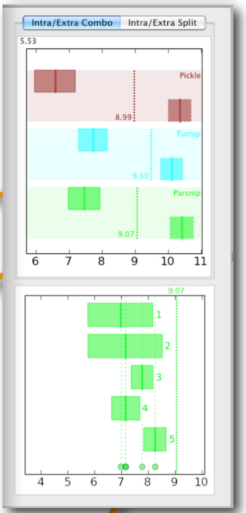
The *Check EGL* button looks through the EGL for hits for each selected gesture and/or example. This can take a long time, so patience is key!

It's important to remember the relationship between the thresholds and the three tabs.

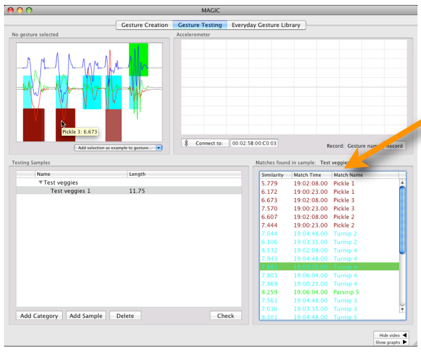
When you adjust the threshold, you affect…



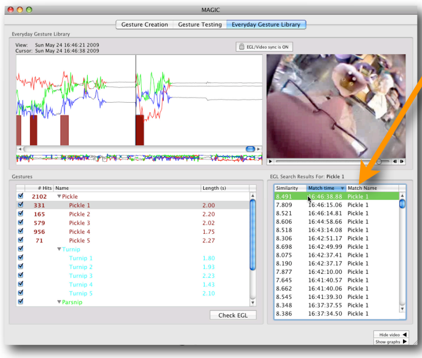
…on the **Creation** tab, the *goodness* value for each gesture example. This, remember, tells you whether a given example matches *all other* examples of its gesture type, and *only* examples of its type.



…on the **Testing** tab, the *matches* to your test samples. If your threshold is too high, some of your test sample gestures won't match. If it's too low, some of the samples may match more than one gesture.



…on the **EGL** tab, the *hits* for each of your examples in the EGL. You want to have as few as possible, but if you make the threshold too low, your gestures might not be recognized at all anymore.



Questions?

If you have any questions, please ask the researcher now.