MAGIC: A Motion Gesture Design Tool

Daniel Ashbrook[1,2] and **Thad Starner**[1]

[1]College of Computing and GVU Center
Georgia Institute of Technology
Atlanta, GA 30332
{anjiro,thad}@cc.gatech.edu

[2]Nokia Research Center Hollywood
Santa Monica, CA 90404
daniel.ashbrook@nokia.com

ABSTRACT

Devices capable of gestural interaction through motion sensing are increasingly becoming available to consumers; however, motion gesture control has yet to appear outside of game consoles. Interaction designers are frequently not expert in pattern recognition, which may be one reason for this lack of availability. Another issue is how to effectively test gestures to ensure that they are not unintentionally activated by a user's normal movements during everyday usage. We present MAGIC, a gesture design tool that addresses both of these issues, and detail the results of an evaluation.

ACM Classification Keywords

H.5.2: User interfaces—*input devices and strategies; prototyping; user-centered design.* D.2.2: Design tools and techniques—*User interfaces*

General Terms

Human Factors

INTRODUCTION

Although often a topic of research [13, 16], motion gestures (as opposed to pen gestures) have not become common outside of gaming systems such as the Nintendo Wii. Research indicates that users prefer devices that are fast to access [3, 14], and motion gestures can provide this desired speed by obviating the need to push 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. We 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 [4]. A number of off-the-shelf tools for experimenting with pattern recognition exist, such as Weka [19] or GT2K [18]; 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 the interaction and make it inconvenient for the user.

With these needs in mind, we next define a list of desiderata for a gesture design tool and discuss related work. We then introduce our system for Multiple Action Gesture Interface Creation (MAGIC), and describe its implementation. We discuss the results of an evaluation of MAGIC's usability, and present related work.

MOTION GESTURE DESIGN TOOL DESIDERATA

In our experience, creating a gesture system has three basic stages. In the first stage, the designer gathers requirements, performs formative user studies and market research, and decides what functionality to consider controlling with gesture. In the second stage, the designer determines how motions by the user will map to functionality activated on the device. She ensures that only intended movements activate functionality, confirms 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 is summative: the designer performs final user testing and deploys the finished product.

In this paper, we discuss the creation of a tool to support the middle stage of design. We 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

CHI 2010, April 10 – 15, 2010, Atlanta, Georgia, USA

responds to mouse clicks, interaction designers should not be required to understand the underlying complexities of gesture recognition in order to build a working system.

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. Tools should encourage iteration by making it easy to explore alternatives. In terms of gesture design, users should be able to quickly try out 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 research in this area thus far [10, 11]. However, designers still need the ability to recall and examine the gestures they've created, understand how they are similar or different from other gestures, and remember what motion corresponds to what function.

support further testing. Outside of recognition concerns, there may be a number of other needs that the designer should consider: social acceptability, memorability, usability in different situations, etc.... 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.

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

RELATED WORK

Our system, MAGIC, is related to *classification* systems, which—although the particulars often differ—share the idea of receiving unknown data and determining what category, or *class*, that data belongs to. Almost all such systems—whether pen [12] gesture, motion [13] gesture, or even face recognition [17]—use a *training and testing* approach. In the training phase, the system is given *labeled data*—a number of examples of data that should be recognized and to which classes the examples belong. In the testing phase, the system is given more data and is asked to guess the correct class; the performance of the system is then estimated by how closely its guesses match the labels.

When providing training examples to a classification system, it is important to ensure that the examples for a given class are similar to one another; that is, the set of examples for a class should be *internally consistent*. Training a speech-based DVD player will work better with many examples of someone saying "play" than if "start" is mixed into the training examples. Similarly, the sets of examples for multiple classes should be *distinguishable* from one another: "stay" is similar enough to "play" that it would be a poor choice for the pause function. Helping the user to understand when and how consistency and distinguishability are compromised is one important consideration for making such systems more usable by non-experts.

A number of classification-based systems in the literature attempt to improve usability; MAGIC was influenced by many of them. One major inspiration is Long's *quill* [12], an interactive system for designing pen gestures. *quill* supports gesture creation by example, and offers automated advice on improving the gestures. In the design process for *quill*, 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 these issues, *quill* includes feedback on the quality of gestures and used language to communicate issues; however, Long states that "many participants did not understand the suggestions," going on to posit that perhaps they "can be made more accessible... by using more diagrams".

Exemplar [6] is a more general system for creating sensor-based interactive systems. In addition to training classifiers, Exemplar can connect to real-world hardware so that interactions can quickly be prototyped. Users of Exemplar train the system by giving it examples of events to be recognized; however, it does not differentiate between testing and actual use, and so does not provide information on performance with respect to internal consistency or between-class differentiability. Such information is critical when designing multiple gestures that should not interfere with one another. Additionally, Exemplar provides no facility for comparing defined events, which may make it difficult to keep track of many examples or many classes.

Exemplar is related to d.tools [7], a system for prototyping interactions with physical devices. d.tools concentrates on reliable (that is, non classifier-based) input, but integrates video analysis of user testing. MAGIC operates in a similar fashion, providing video linked to input, to help the system designer recall what input was provided for different actions.

Crayons [4] is an interactive computer vision classifier training system that takes in pre-recorded images and allows users to interactively specify which classes various parts of an image should belong to. 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.

EnsembleMatrix [15] 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.

Wobbrock *et al.* [20] investigated gestures from a different angle, using a tabletop environment to elicit motions from users in response to a variety of graphical commands such as "move", "zoom" or "rotate". The authors found several commonalities in mental models about some of the commands. This study gives a good model for designers to follow during the initial formative phase of gesture system creation.

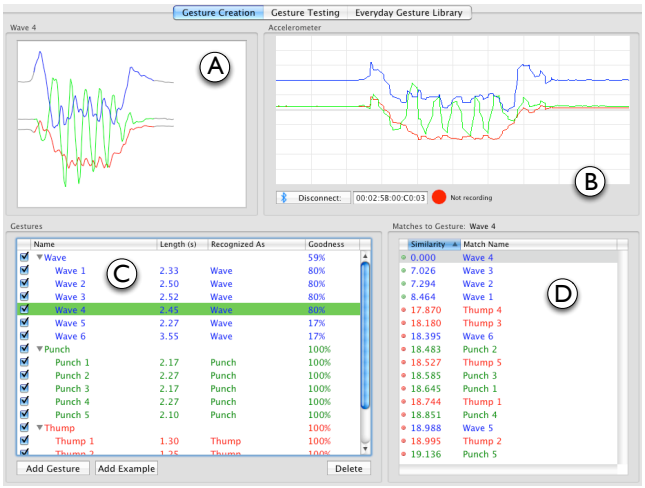

Figure 1. The Gesture Creation tab. A: recorded gesture view; B: live sensor view; C: list of gestures and gesture examples; D: sorted list of distances from currently selected example to every other example.

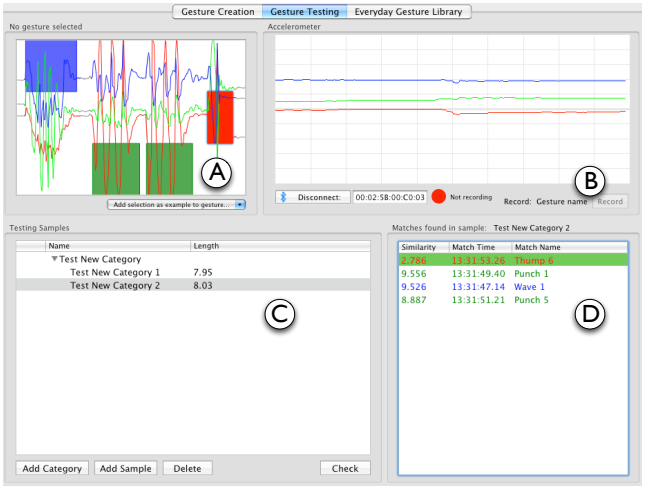

Figure 2. The Gesture Testing tab. A: recorded test sample view with boxes highlighting matches; B: live sensor view; C: list of test samples; D: list of gestures matching currently selected test sample.

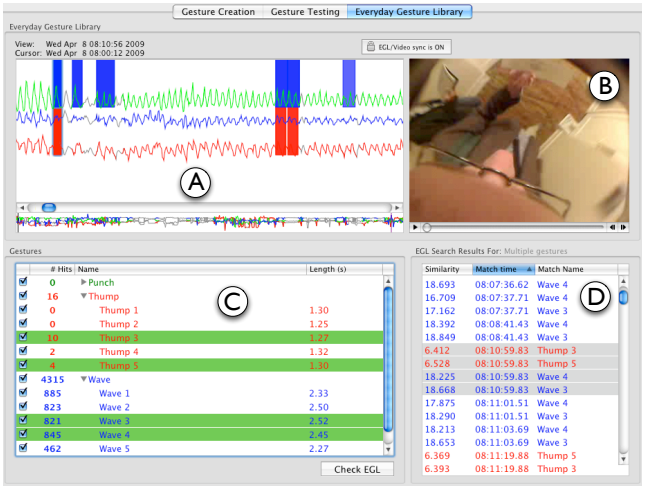

Figure 3. The Everyday Gesture Library tab. A: EGL view with boxes highlighting gesture occurrences; B: EGL video synchronized to EGL view; C: gestures with number of occurrences in EGL (# Hits); D: list of occurrences in EGL for selected gestures.

MAGIC: A GESTURE DESIGN TOOL

MAGIC is partially inspired by the design/test/analyze model developed by Klemmer *et al.* [9] and supports a similar three-stage workflow. The stages are not strictly linear; a designer will fluidly move between them as they work. MAGIC reflects these stages with three tabs in the interface: *Gesture Creation*, *Gesture Testing*, and *Everyday Gesture Library* (see Figures 1, 2, and 3). The stages 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. 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 1) includes support for creating gesture classes, recording examples, and understanding how examples and classes relate to each other in terms of recognition performance (*consistency* and *distinguishability*). To support experimentation, gestures and examples may be deleted or temporarily disabled.

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 2) 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 eliminate 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 3) 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 move-

ments. This pre-recorded data is called the "Everyday Gesture Library" (EGL).

SENSING AND GESTURE RECOGNITION

MAGIC was designed to accommodate a variety of sensors. For prototyping purposes, we use a wrist-mounted Bluetooth $\pm$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. We chose to mount the accelerometer on the wrist, which has been shown to be a location that is easily accessible [1] for control and display and can allow for hands-free operation of devices.

MAGIC can also take advantage of a variety of gesture recognition algorithms. Due to its ease of implementation and relative computational efficiency (when optimized) we chose to use dynamic time warping (DTW). Given two signals, DTW returns a "distance" between them. For more information on the particular methods used in MAGIC, see Fu *et al.* [5].

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 fast Fourier transform (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 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.

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*.

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. Our research group has over a decade's experience with gesture interfaces for everyday life [8, 13, 18] and has struggled with this issue. One solution is to test each new gesture in the field as it is designed. 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.

MAGIC offers a partial solution to this problem. Rather than requiring user testing in the initial phases of gesture design, MAGIC allows designers to iterate rapidly 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"—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. MAGIC performs a windowed search over the EGL to find occurrences, which indicate times 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.

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.

In both the creation and testing phases, the designer records motions. MAGIC displays a continuously-updating graph of the output from the accelerometer (Figure 1B). The x, y, and z axes are displayed as red, green, and blue lines, respectively. After an example has been recorded, it is displayed in the same way (Figure 1A). 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 5). 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 both representations of 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. This functionality is similar to that of d.tools [7], which allows designers to replay video and sensor readings from a user test session.

Visualization: Gesture Design

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. Because confusion matrices can be difficult

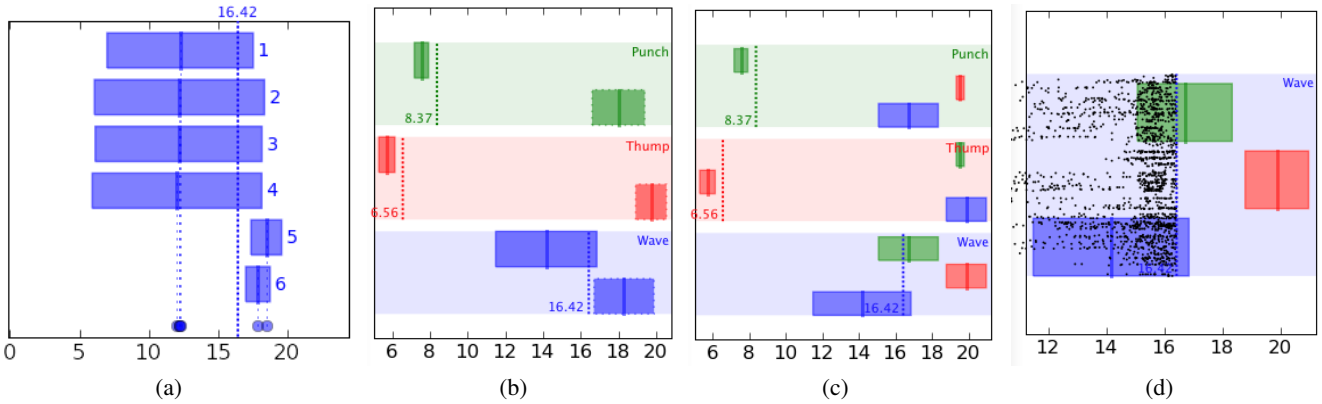

Figure 4. Mean and standard deviation distance graphs for (a) each example within a class, (b) each class versus all other classes in aggregate, (c) each class versus all other classes individually, and (d) a magnified view of a single row of (c), with dots representing occurrences in the EGL. The horizontal scale in each case represents the DTW distance.

to read for non-experts [12], MAGIC provides different visualizations derived from the same information. Confusion matrices are constructed by comparing every example in every class with every other example in every class. MAGIC displays this information on an example-by-example basis in the *match list* (Figure 1D). 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 1D, 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.

MAGIC provides a summary of the match list by calculating a "goodness" value for each example. An example's goodness is based on the precision and recall for cross-validation (the example compared to every other example in all classes). *Precision* is the percentage of results labeled as a class that actually belong to the class; in *Wave 4*'s match list, it is the percentage of items with a green dot that actually belong to class *Wave*. *Recall* is the percentage of a class as a whole that was labeled as belonging to the class; in *Wave 4*'s match list, it is is the percentage of *Wave* examples that have a green dot. The goodness (G), is the harmonic mean of precision (P) and recall (R): $G = 2 \cdot (P \cdot R)/(P + R)$. 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. 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.

Figure 4(a) is an example of a graph representing internal consistency for the class *Wave*. Each numbered bar represents a single example in the class; in this case, six 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 feature allows the overall distribution of distances to 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 adjust the threshold interactively. By looking at the graph, the designer is able to identify outliers visually. In this case, two examples—5 and 6—are quite different from the other examples, and might cause a low goodness score.

Figure 4(b)–(d) shows graphs visualizing inter-class variability for each class as compared to all other classes. In Figure 4(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 4(c) is similar to Figure 4(b), but splits the dotted-line box into its constituent classes. While the graph in 4(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 4(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 4(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.

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 de-

signer 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 2D and 3D), 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 2A and 3A). 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 3A). 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 5(b), 5(c), 3B) 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 of the problem. MAGIC displays each match in the EGL as a black dot on the inter-class graphs. This is illustrated for the *Wave* class in Figure 4(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 4(d), the threshold cannot be moved to the left far enough to eliminate all EGL matches.)

EVALUATION

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

- qualitatively assess the usability of MAGIC by observing users utilizing it to design gestures;
- understand the design strategies used by designers when creating gestures; and
- 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.

EGL Creation

In order to create an Everyday Gesture Library to test against, eight volunteers wore data collection systems for a total of over 58 hours over a period of seven days. The data collection system consisted of the wrist-mounted accelerometer, an Asus eeePC 901 netbook computer, a shoulder-mounted bag, and a hat with a fisheye camera lens mounted in the brim, pointing downwards (see Figure 5(b) for a representative image). The EGL includes a wide variety of activities, such as eating, working at the computer, shopping at a market, cooking, attending an academic conference, walking to work, attending meetings, driving, playing pool, and napping. A small portion (about 5.3 hours) of collected data was separated for use in the experiment, with the remaining data reserved for *post-hoc* testing.

Procedure

There were two experimental groups, with the same task for each group. Group *noEGL* received the MAGIC interface with the EGL tab removed, and participants in this group were ignorant of the existence of the EGL; group *EGL* used the full interface with the experimental portion of the EGL.

Each participant was seated at a desk in a chair without arms (to allow for free arm movement). To begin, the participant was requested to wear the hat with the camera, and to wear the wireless accelerometer on the left wrist. Each participant worked through a tutorial on the use of MAGIC which also provided a brief introduction to gesture recognition. On average, participants required 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 design and create eight gestures to control a hypothetical wrist-mounted, gesture-controlled digital audio player. The functions to control were: Play/Pause, Next Track, Previous Track, Volume Up 10%, Volume Down 10%, Next Playlist, Previous Playlist, and Shuffle. These functions were chosen because they are commands that would be plausible to control with gestures, and were intended for a type of device with which participants would be familiar.

In addition, the instructions asked participants to ensure that each gesture met the following criteria:

- 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. 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, comprised of the Questionnaire for User Interface Satisfaction (QUIS) [2], QUIS-inspired questions about the gestures, and a section requesting descriptions of why the participant chose the particular movements for each gesture.

Participants

We recruited a total of 16 participants; one participant took over two hours to complete the tutorial and was therefore

(a) (b) (c)

Figure 5. Experiment setup, showing participant wearing accelerometer and hat camera (a) and views from hat camera (b) and monitor camera (c).

discarded as an outlier, leaving 15 participants. Our criteria for recruitment 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 our institution.

We requested participants to complete a brief demographic survey before starting the experiment. The questions included experience with motion-sensing devices such as the Nintendo Wii game system, and the iPhone or T-Mobile G1 mobile phones and, on a 9-point scale, experience with user-centered design, with designing user interfaces, and with pattern recognition. See Table 1 for a summary of participant information.

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	3.4	1.4	6.6	5.6	3.0

Table 1. 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).

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." Participant response to the QUIS questionnaire was positive, although not overwhelmingly so, with the mean response for nearly every question falling between 6 and 8 on a scale of 1–9.

User Performance

On average, participants required two hours to complete the experimental task, regardless of the condition. Contrary to our expectations, almost all participants proceeded through the task in a very linear manner, creating all of the classes and examples first, then testing them on the Testing or EGL tabs, rather than creating one gesture at a time and testing each one. 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.

Overall, participants were successful in the gesture creation task, with none failing to create distinct gestures for the eight functions. The mean goodness value for the gestures over all the participants was 89.8% ($SD = 15.9\%$), 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.

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.

Retrospection

During post-experiment interviews, it was the features enabling retrospection which were identified as most useful, especially the video playback of example gestures. Often, participants forgot how they had made a particular gesture and used the video to remind them of the proper movements.

The recorded video also proved useful during initial gesture creation. 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 per-

formed better than the others, the participant incorporated the unintentional movement into the gesture, re-recording the other examples to include the motion.

The video automatically saved 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 the correct gesture class—and the participant would watch the video to determine 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 5, participants were recorded by both a hat-mounted and a monitor-mounted camera during gesture recording. Contrary to our expectations, the monitor camera was more popular amongst participants than was the hat camera. We 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 1A). 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. This 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 the visualization currently used in most systems involving interaction with time-series data [6, 18].

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. Overall, we 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 1B) useful, especially for experimenting with the effects of arm movements on the resulting shape of the trace. Contrary to our 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, including the "Recognized As" and "Goodness" columns (Figure 1C), the match box showing similarity scores (1D), and the intra-gesture graph (4(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 4(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.

Everyday Gesture Library

In this section, we assume $p < .05$ for significance. As expected, the participants with access to the EGL (condition *EGL*) had fewer EGL occurrences—disregarding an outlier, an average of 1.9 per gesture per hour (SD = 9.6)—than those without (condition *noEGL*), who had an average of 52.1 occurrences per gesture per hour (SD = 117.8). Four participants (one from condition *noEGL*, three from *EGL*) achieved zero occurrences in the EGL. Four had an average of one occurrence/gesture/hour, one had four occurrences/gesture/hour, and the rest of the participants had more than five occurrences/gesture/hour.

As mentioned earlier, eight volunteers collected EGL data, and about five hours of the data was used during the experiment. An independent-samples t-test on the number of occurrences in the reserved portion of the EGL revealed a significant difference between the two conditions. Discarding an outlier subject (with number of occurrences $>$ 2SD), there was a significant 87% correlation between the number of occurrences in the experimental portion of the EGL and the remaining 53 hours, suggesting that—at least for this particular combination of sensor, recognition algorithm, and task—that a small EGL can allow for fast and accurate preliminary testing before moving on to a larger one.

The EGL proved a success in terms of avoiding gestures that might appear in everyday life. In terms of this goal, participants were positive about the utility of the EGL and the usability of the interface. However, the difficulty of finding gestures that did not occur in the EGL caused some users frustration. In the words of one condition *EGL* participant, "I just kinda feared the EGL."

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. One participant commented, "I didn't care *why* I was hitting the Everyday Gesture Library; I can't change what's in there!"

Design Strategies

In designing gestures, participants had several competing concerns. For condition *noEGL*, we 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.

Condition *EGL* 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.

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, we 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 common motion created by 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 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.

DISCUSSION

Earlier, we introduced a number of desiderata for a motion-gesture design tool. We now revisit these items and discuss how well MAGIC fulfills them.

allow non-expert use We consider MAGIC to be a success on this point. As can be seen in Table 1, our participants were not expert in pattern recognition. The three participants who rated themselves as expert or near-expert in pattern recognition did appear to better understand the intra/extra graphs; this result was expected, as those graphs present familiar information such as inter- and intra-class variance.

allow expert use Some facilities exist in MAGIC for a user more expert in pattern recognition to adjust system behavior: users can pick and choose between examples to use, and can adjust the recognition threshold for any gesture. We plan to add further capabilities: for example, we could expose the parameters used in DTW, or give different options for calculating features. It was clear during the study, however, that the task was difficult enough that even the users claiming pattern recognition expertise did not change the parameters in any way, preferring to adjust their gestures to the system.

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.

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 can also be used for training others to use the gestures, or to test for others' reactions in terms of social acceptability.

Shortcomings

There are a number of shortcomings in MAGIC that we plan to address. The difficulty experienced by some participants in understanding the intra- and inter- graphs (Figure 4) 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 we believe there is value in communicating it.

While effective for our experiment, we wish to improve our gesture recognition algorithms. By using new or different features, or implementing more complex algorithms such as hidden Markov models [18], MAGIC should be more effective in recognizing intentional gestures and ignoring unintentional ones. We also plan to integrate other sensors, such

as gyroscopes for more accurate motion sensing, or microphones for an entirely different modality of operation.

As many of our 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 we are aware, no such guidelines exist. However, given the data we now possess, we intend to investigate creating an automated gesture advisor, not unlike Long's *quill* system for pen gestures [12].

Lacking specific guidelines on motions, we *can* provide automated behavior in other areas. For example, very few participants manually adjusted the threshold—preferring the default automatic behavior—and 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 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.

While we feel that our experiment accomplished our goals to assess MAGIC's usability, understand users' design strategies, and determine the efficacy of the EGL, our evaluation was lacking in some areas due to its preliminary nature. A primary area we will improve upon in our next study 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 we wanted for them to achieve expert status with the system. We plan to divide the study over multiple days to understand how users interact with the software longer term.

Another issue with our study is the lack of variability in gesturing situations. All of our participants were seated during gesture creation and testing, while the EGL was recorded in many different mobility situations. We plan to make a mobile variant of the data collection system to allow designers to move about as they record gestures.

CONCLUSION

We have presented MAGIC, an interactive system for exploring and designing motion gestures. MAGIC encourages iteration in design, provides facilities for retrospection of input, and allows the designer to test the created gestures against a corpus of everyday activity to ensure that the gestures will not be unintentionally activated by a user.

References

1. D. Ashbrook, J. Clawson, K. Lyons, N. Patel, and T. Starner. Quickdraw: The impact of mobility and on-body placement on device access time. In *Proc. CHI*, 2008.
2. J. Chin, V. Diehl, and K. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. In *Proc. CHI*, 1988.
3. Y. Cui, J. Chipchase, and F. Ichikawa. A cross culture study on phone carrying and physical personalization. In *HCI International*, 2007.
4. J. Fails and D. Olsen. A design tool for camera-based interaction. In *Proc. CHI*, 2003.
5. A. W.-C. Fu, E. Keogh, L. Y. H. Lau, C. A. Ratanamahatana, and R. C.-W. Wong. Scaling and time warping in time series querying. In *The VLDB Journal*, 2008.
6. B. Hartmann, L. Abdulla, M. Mittal, and S. Klemmer. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proc. CHI*, 2007.
7. B. Hartmann, S. Klemmer, M. Bernstein, L. Abdulla, B. Burr, A. Robinson-Mosher, and J. Gee. Reflective physical prototyping through integrated design, test, and analysis. In *Proc. UIST*, Oct 2006.
8. J. Kim, J. He, K. Lyons, and T. Starner. The gesture watch: A wireless contact-free gesture based wrist interface. In *Proc. ISWC*, Jan 2007.
9. S. Klemmer, A. Sinha, J. Chen, J. Landay, N. Aboobaker, and A. Wang. Suede: a wizard of oz prototyping tool for speech user interfaces. In *Proc. UIST*, 2000.
10. S. Kratz and R. Ballagas. Unravelling seams: Improving mobile gesture recognition with visual feedback techniques. In *Proc. CHI*, 2009.
11. J. Linjama, P. Korpipää, J. Kela, and T. Rantakokko. Actioncube: a tangible mobile gesture interaction tutorial. In *Proc. TEI*, 2008.
12. C. A. Long. *Quill: a Gesture Design Tool for Pen-based User Interfaces*. PhD thesis, University of California, Berkeley, 2001.
13. T. Starner, J. Auxier, D. Ashbrook, and M. Gandy. The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In *Proc. ISWC*, Atlanta, GA, 2000.
14. T. Starner, C. M. Snoeck, B. Wong, and R. M. McGuire. Use of mobile appointment scheduling devices. In *Proc. CHI*, 2004.
15. J. Talbot, B. Lee, A. Kapoor, and D. Tan. Ensemblematrix: Interactive visualization to support machine learning with multiple classifiers. In *Proc. CHI*, 2009.
16. B. Tognazzini. The "starfire" video prototype project: a case history. *Proc. CHI*, pages 99–105, 1994.
17. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
18. T. Westeyn, H. Brashear, A. Atrash, and T. Starner. Georgia tech gesture toolkit: supporting experiments in gesture recognition. In *Proc. ICMI*, 2003.
19. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.
20. J. Wobbrock, M. R. Morris, and A. D. Wilson. User-defined gestures for surface computing. In *Proc. CHI*, 2009.