Journal of Machine Learning Research 14 (2013) 209-242 Submitted 10/11; Revised 6/12; Published 1/13

MAGIC Summoning: Towards Automatic Suggesting and Testing of Gestures With Low Probability of False Positives During Use

Daniel Kyu Hwa Kohlsdorf DKOHL@TZI.DE
Thad E. Starner THAD@CC.GATECH.EDU
GVU & School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332

Editors: Isabelle Guyon and Vassilis Athitsos

Abstract

Gestures for interfaces should be short, pleasing, intuitive, and easily recognized by a computer. However, it is a challenge for interface designers to create gestures easily distinguishable from users' normal movements. Our tool MAGIC Summoning addresses this problem. Given a specific platform and task, we gather a large database of unlabeled sensor data captured in the environments in which the system will be used (an "Everyday Gesture Library" or EGL). The EGL is quantized and indexed via multi-dimensional Symbolic Aggregate approXimation (SAX) to enable quick searching. MAGIC exploits the SAX representation of the EGL to suggest gestures with a low likelihood of false triggering. Suggested gestures are ordered according to brevity and simplicity, freeing the interface designer to focus on the user experience. Once a gesture is selected, MAGIC can output synthetic examples of the gesture to train a chosen classifier (for example, with a hidden Markov model). If the interface designer suggests his own gesture and provides several examples, MAGIC estimates how accurately that gesture can be recognized and estimates its false positive rate by comparing it against the natural movements in the EGL. We demonstrate MAGIC's effectiveness in gesture selection and helpfulness in creating accurate gesture recognizers.

Keywords: gesture recognition, gesture spotting, false positives, continuous recognition

1. Introduction

The success of the Nintendo Wii, Microsoft Kinect, and Google's and Apple's mobile devices demonstrates the popularity of gesture-based interfaces. Gestural interfaces can be expressive, quick to access, and intuitive (Guimbretière and Winograd, 2000; Pirhonen et al., 2002; Starner et al., 1998; Witt, 2007). Yet gesture-based interfaces may trigger functionality incorrectly, confusing normal movement with a command. For example, the Apple iPod's "shake-to-shuffle" gesture, which is intended to signal when the user wants to skip a song and randomly select another, tends to trigger falsely while the user is walking (see Figure 1a). Part of the difficulty is that the recognizer must constantly monitor an accelerometer to determine if the gesture is being performed. Some accelerometer or gyro-based interfaces constrain the problem by requiring the user to segment the gesture by pressing a button. For example, in Nintendo's Wii Bowling the player presses the "B" trigger when beginning to swing his arm and releases the trigger at the end of the swing to release the virtual bowling ball. Such a push-to-gesture approach is similar to the push-to-talk method that speech recognition researchers use to improve performance. Yet such mechanisms can slow interactions, confuse users, and limit the utility of gesture interaction. For example, the fast, easy-to-access nature of the shake-to-shuffle gesture would be impeded if the user needed to hold a button to perform the gesture. Ideally, such free-space "motion gestures" (Ashbrook, 2009) should be short, pleasing to perform, intuitive, and easily recognized by a computer against a background of the user's normal movements.

Touchpad gesture shortcuts, which upon execution can start an affiliated application on a laptop or mobile phone (Ouyang and Li, 2012), are another example of command gestures that must be differentiated from everyday motions. Fortunately, these gestures are naturally isolated in time from each other since most touchpad hardware does not even provide data to the operating system when no touches are being sensed. However, an interface designer must still create gesture commands that are not easily confused with normal click or drag and drop actions (see Figure 1b).

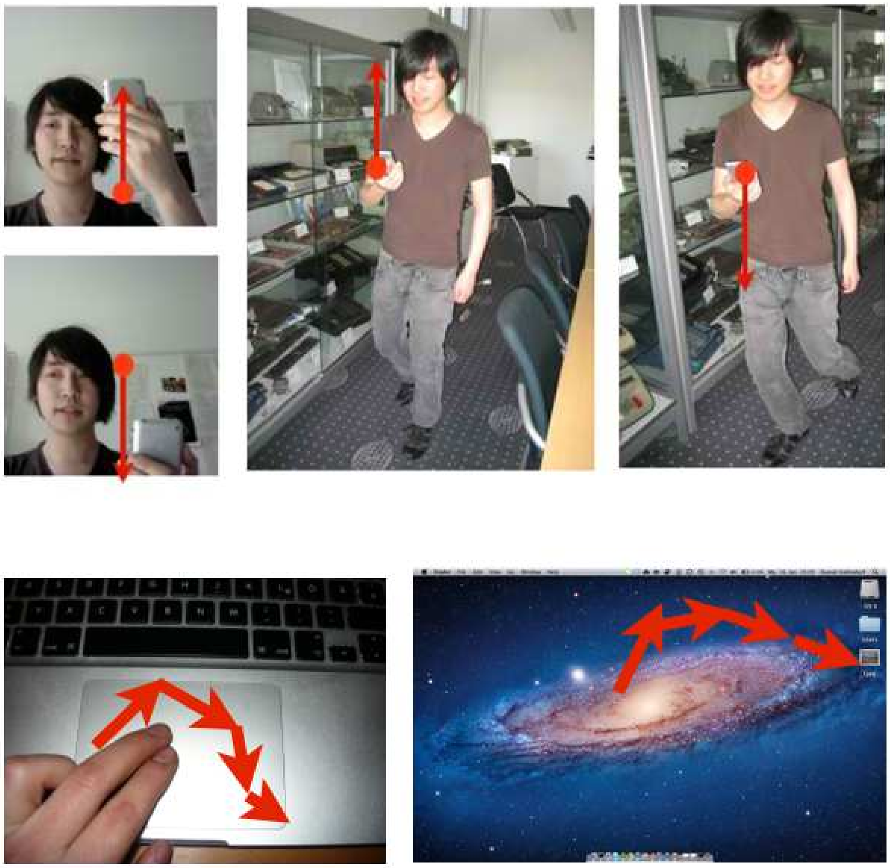


Figure 1: *Top:* A "shake-to-shuffle" gesture (*left*) can be confused with normal up-and-down movement while walking (*right*). *Bottom:* A touchpad shortcut gesture (*left*) can be confused with normal cursor movement (*right*).

Many "direct manipulation" (Hutchins et al., 1985) gestures such as pointing gestures and pinch-to-zoom gestures are used in modern interfaces. These gestures provide the user continuous feedback while the gesture is occurring, which allows the user to adjust to sensing errors or cancel the interaction quickly. However, representational gestures that are intended to trigger a discrete action are less common. We posit that their relative scarcity relates to the difficulty of discovering appropriate gestures for the task. Our previous studies have shown that designing command gestures that do not trigger accidentally during normal, everyday use is difficult for both human computer interaction (HCI) and pattern recognition experts (Ashbrook and Starner, 2010). In addition, the current process to determine the viability of a gesture is challenging and expensive. Gestures are often found to be inappropriate only after the system has entered user testing. If a gesture is found to trigger accidentally during testing, the gesture set has to be changed appropriately, and the testing has to be repeated. Such an iterative design cycle can waste a month or more with each test. Thus, we posit the need for a tool to help designers quickly judge the suitability of a gesture from a pattern recognition perspective while they focus on the user experience aspects of the gestural interface.

Several gesture design tools have been described in the HCI literature (Dannenberg and Amon, 1989; Long, 2001; Fails and Olsen, 2003; Maynes-Aminzade et al., 2007; Dey et al., 2004), yet none address the issue of false positives. Similarly, most gesture recognition toolkits in the pattern recognition and related literature focus on isolated gestures (Wobbrock et al., 2007; Lyons et al., 2007) or the recognition of strings of gestures, such as for sign language (Westeyn et al., 2003). Rarely do such tools focus on gesture spotting (Yang et al., 2009) for which the critical metric is false positives per hour.

Ashbrook and Starner (2010) introduced the Multiple Action Gesture Interface Creation (MAGIC) Toolkit. A MAGIC user could specify gesture classes by providing examples of each gesture. MAGIC provided feedback on each example and each gesture class by visualizing intra- and interclass distances and estimating the prototype recognizer's accuracy by classifying all provided gesture examples in isolation. Unlike the above tools, MAGIC could predict whether a query gesture would tend to trigger falsely by comparing the gesture to a database of movements recorded in the everyday lives of users. Primarily designed as an HCI Tool, the system used a nearest neighbor method with a dynamic time warping (DTW) distance measure (Fu et al., 2008).

One shortcoming of this work was that the relative false positive rates predicted in user studies were not compared to the actual false positive rates of a gesture recognizer running in the field. Another shortcoming was the long time (up to 20 minutes) needed to search for potential hits in a database of everyday user movements (an "Everyday Gesture Library" or EGL) even while using approximations like scaling with matching (Fu et al., 2008). MAGIC was designed as an interactive tool, yet due to the delay in feedback, gesture interaction designers waited until all gestures were designed before testing them against the EGL. Often, when doing an EGL test in batch, the interface designers discovered that many of their gestures were poor choices. Designers "learned to fear the EGL." Faster feedback would allow designers to compare candidate gestures to the EGL as they perform each example, speeding the process and allowing more exploration of the space of acceptable gestures. Another result from previous studies is that users were frustrated by encountering too many false positives in the Everyday Gesture Library (Ashbrook and Starner, 2010). In other words, many designed gestures are rejected since the number of predicted false positives is too high.

Here, we focus on the pattern recognition tasks needed to create MAGIC Summoning, a completely new, web-based MAGIC implementation designed to address the needs discovered from using the original. Section 2 introduces the basic operation of the tool. Section 3 describes an indexing method for the EGL using a multi-dimensional implementation of indexable Symbolic Aggregate approXimation (iSAX) that speeds EGL comparisons by an order of magnitude over the DTW implementation. While not as accurate as DTW or other methods such as HMMs, our system's speed allows interface designers to receive feedback after every gesture example input instead of waiting to test the gesture set in batch. We compare the iSAX approach to linear searches of the EGL with HMMs and DTW to show that our approach, while returning fewer matches, does predict the relative suitability of different gestures. Section 4.4 continues this comparison to show that the predictions made by MAGIC match observations made when the resulting gesture recognizers are tested in a real continuous gesture recognition setting. Sections 5 and 6 provide additional details. The first describes a method of using the EGL to create a null (garbage) class that improves the performance of a HMM classifier and a DTW classifier when compared to a typical thresholding method. The second demonstrates the stability of our method by examining its sensitivity to its parameters and provides a method capable of learning reasonable defaults for those parameters in an unsupervised manner. These sections expand significantly upon previous work published in Face and Gesture (Kohlsdorf et al., 2011), while the remaining sections represent unpublished concepts.

Section 7 may be of the most interest to many readers. This section describes how MAGIC Summoning suggests novel gestures that are predicted to have a low probability of false positives. While the capability may be surprising at first, the technique follows directly from the iSAX indexing scheme. In Section 7.2 we show that the suggested gestures have low false positive rates during a user study in a real life setting. In our tests, the space of gestures that are not represented in EGLs tends to be large. Thus, there are many potential gestures from which to choose. Section 7.3 describes our attempts at finding metrics that enable ordering of the suggested gestures with regard to brevity, simplicity, and "quality."

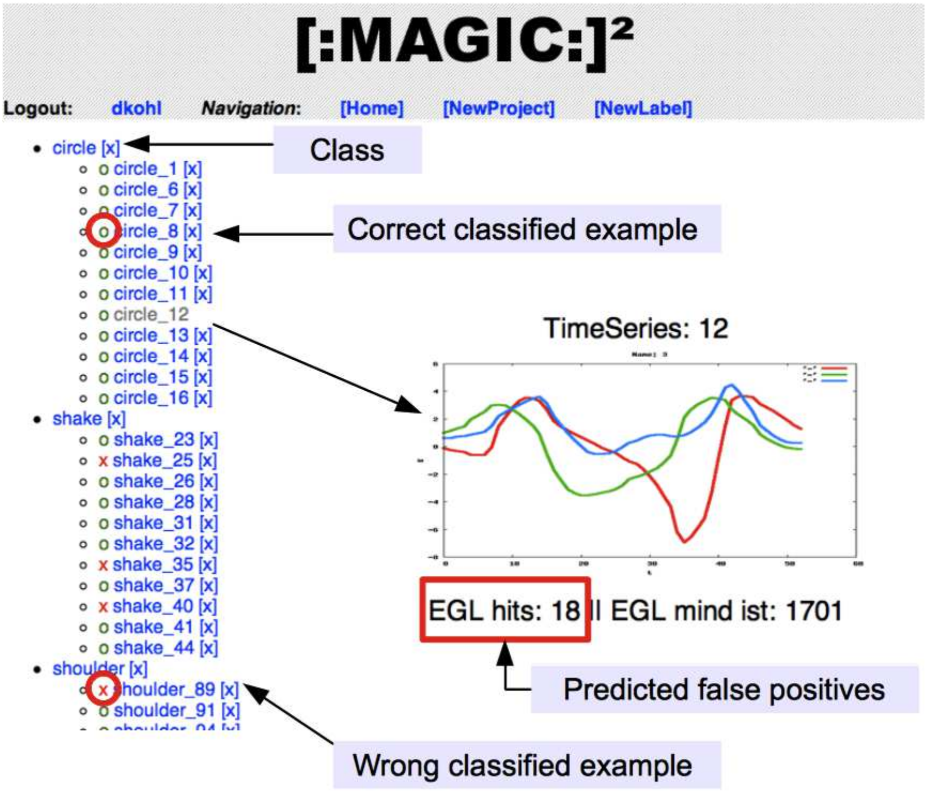
2. MAGIC Summoning Web-based Toolkit

MAGIC Summoning is a web-based toolkit that helps users design motion-based gestural commands (as opposed to static poses) that are expected not to trigger falsely in everyday usage (Kohlsdorf, 2011; Kohlsdorf et al., 2011). All MAGIC experiments described in this paper focus on creating **user-independent** recognizers. This choice reflects our interest in creating useful gesture interfaces and is also due to practicality; collecting large data sets for the EGL from a single user is time consuming and onerous. To ground the discussion with a practical problem, we focus on the challenge of designing gestures performed by moving an Android phone in one's hand. We assume a three-axis accelerometer, which is always included in modern Android phones. The goal is to create gestures (and an appropriate classifier) that, when recognized, trigger functions like "open mailbox" or "next song." Without a push-to-gesture trigger, such gestures are highly susceptible to false positives (Ashbrook, 2009), which emphasizes the need for the MAGIC tool.

2.1 Creating Gesture Classes And Testing For Confusion Between Classes

MAGIC Summoning has two software components: a gesture recorder running on the Android device and the MAGIC web application. The first step in gesture creation is to start a new project in the web service. The interface designer specifies the set of gestures through collecting training data for each of the gestures using the recorder. In order to record a training example, the interaction designer opens the recorder on his smart phone and performs the gesture. The recorder automatically estimates when the gesture starts and when it ends using the method described by Ashbrook (2009). Specifically, the recorder tracks the variance of the accelerometer data in a sliding window. If the variance is above a user-defined threshold, recording starts. If it falls below the threshold, then recording ends.

After the example is recorded, the designer is asked to associate the example with an appropriate gesture label, and the recorder uploads the example to the web. The designer evaluates the gesture in the web application to determine how well it can be distinguished from other gestures. All gestures and their examples are listed in MAGIC Summoning's sidebar (see Figure 2). Examples marked with a red cross are misclassified given the current model, and instances marked with a green circle indicate correct classification. By default, MAGIC Summoning uses a one nearest neighbor classifier with dynamic time warping (NN-DTW) to classify gestures, although other classifiers such as a hidden Markov model (HMM) could be substituted. By clicking on an instance, the designer can see the raw sensor data plotted for that example as well as the predicted number of false positives in the EGL (the method used to calculated this number is explained in Section 3).


[:MAGIC:]²
Logout: dkohl Navigation: [Home] [NewProject] [NewLabel]
Class
Correct classified example
TimeSeries: 12
Predicted false positives
Wrong classified example

- circle [x]
  - o circle_1 [x]
  - o circle_6 [x]
  - o circle_7 [x]
  - o circle_8 [x]
  - o circle_9 [x]
  - o circle_10 [x]
  - o circle_11 [x]
  - o circle_12
  - o circle_13 [x]
  - o circle_14 [x]
  - o circle_15 [x]
  - o circle_16 [x]
- shake [x]
  - o shake_23 [x]
  - x shake_25 [x]
  - o shake_26 [x]
  - o shake_28 [x]
  - o shake_31 [x]
  - o shake_32 [x]
  - x shake_35 [x]
  - o shake_37 [x]
  - x shake_40 [x]
  - o shake_41 [x]
  - o shake_44 [x]
- shoulder [x]
  - x shoulder_89 [x]
  - o shoulder_91 [x]

EGL hits: 18 || EGL mind ist: 1701

Figure 2: Magic Summoning showing the gesture classes, their examples, and the number of EGL hits (lower numbers are better).

Clicking on a gesture in the sidebar opens a view with statistics about it. One statistic is the goodness of the gesture. The goodness is defined as the harmonic mean of precision and recall (Ashbrook, 2009):

$$goodness = 2 * \frac{precision * recall}{precision + recall}.$$

Similar to the original work by Ashbrook (2009), MAGIC Summoning provides users with information about the inter-class distance and the intra-class distance of the gesture. Both are visualized using a mean and standard deviation plot. In an intra-class distance plot we calculate the means and standard deviations of the distances from all examples in a class to all other examples in that class and visualize the result as a box plot (see Figure 3). In an inter-class distance plot we calculate the means and standard deviations from one class to all the others in the training set. The distance between two classes is the mean distance of all examples of one class to all examples of another. These statistics and visualizations help designers find inconsistencies in the examples of a given gesture class as well as unintentional similarities between classes.

2.2 Android Phone Accelerometer Everyday Gesture Library

We collected a large EGL ($>$ 1.5 million seconds or 19 days total) using six participants' Android phones in Bremen, Germany. The age of the participants ranged from 20 to 30 years. We implemented a background process that wrote the three-axis accelerometer data to the phone's flash memory. Unfortunately, the sampling frequency varied as the models of Android phones we used return samples only when the change in the accelerometer reading exceeds a factory-defined threshold. The phones used are the Motorola Droid, the Samsung Galaxy, HTC Nexus One, the HTC Legend, and the HTC Desire. Other EGLs loadable in MAGIC include movements sensed with a Microsoft Kinect and gestures made on trackpads. We focus mostly on our EGL created with Android phones, but readers interested in experiments with other sensors can refer to Kohlsdorf (2011) for more information.

2.3 Testing For False Positives With The EGL

The original Macintosh-based MAGIC tool displayed a timeline that showed which candidate gesture matched the EGL and at which time. However, gesture designers did not care when or why a given gesture showed a given false positive in the EGL; they just wished to know how many "hits" occurred in the EGL so that they could accept or reject the gesture (Ashbrook, 2009). Thus, we omitted the timeline for simplicity in the web-based application. In the following section we will describe our accelerated method for testing a gesture for potential false positives against the EGL. This method enables rapid iteration on different gesture sets by the interaction designer.

If a user is displeased by the results after testing, he can delete gestures suspected of high false positive rates or misclassification errors and design new gestures. When the user is satisfied with the gesture set, MAGIC Summoning can train a classifier based on hidden Markov models (HMMs) or the default NN-DTW method. The user can then download the trained recognizer. Note that we do not suggest using the iSAX method used to search the EGL as a gesture recognizer as we have tuned the method for speed, not accuracy.

3. False Positive Prediction

When testing a gesture set against the EGL, the original MAGIC calculates the DTW distance for every example of each candidate gesture, sliding a window through time across the EGL and allowing the window to grow or shrink to better match the example when a potential close match is discovered. If the resulting distance is above a certain user-defined threshold it counts as a false

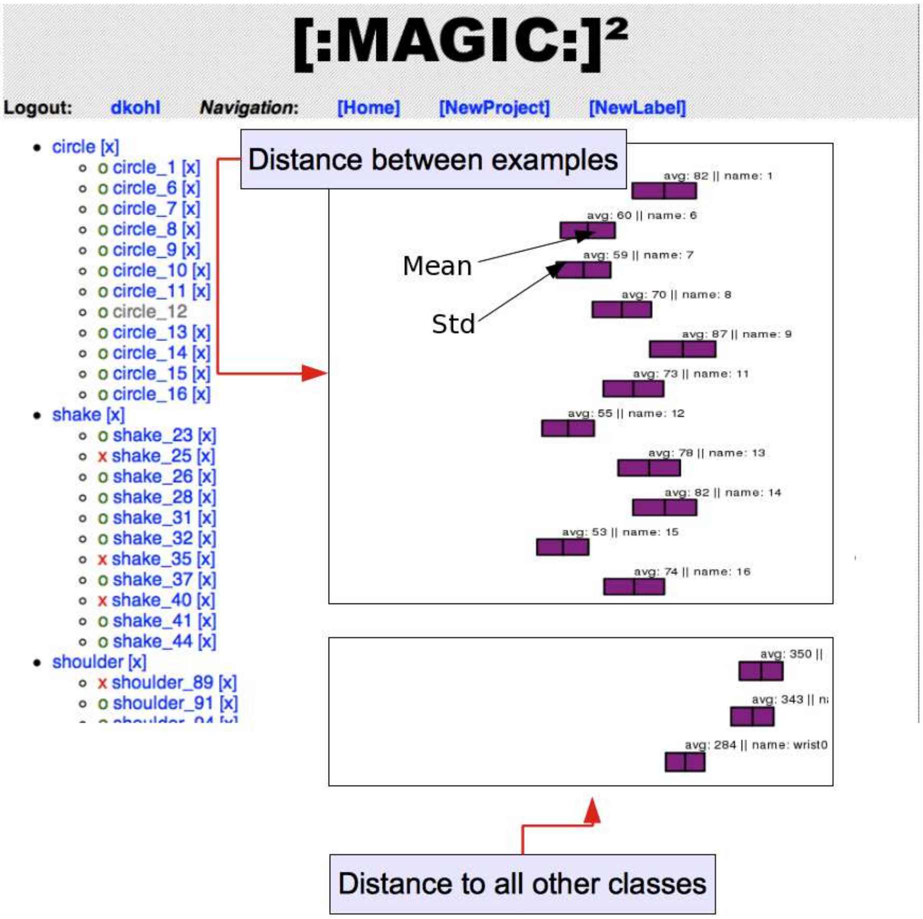


Figure 3: Mean and standard deviation of the distance between each example in a class and of the class as a whole in relation to other classes.

positive "hit." Ashbrook and Starner (2010) assert that the sum of the hits predicts how well the gesture will perform in everyday life (an assertion supported by our experiments described later).

In optimizing the speed of the EGL comparison, Ashbrook (2009) observed that not all regions of the EGL need checking. Since we are interested in motion-based gestures instead of static poses, parts of the EGL with low variance in their signal need not be examined. Thus, we pre-process the EGL to find "interesting" regions where the average variance over all dimensions in the sensor data in a region defined by a sliding window over 10 samples exceeds a given threshold (see Figure 4).[1] Eliminating regions from the EGL that can not possibly match candidate gestures significantly speeds EGL search. Note that a similar technique was described earlier to segment gestures when the interface designer is creating examples of candidate gestures. All experiments in this paper will use these techniques.

[1] Word spotting algorithms in speech recognition perform similar checks, rejecting regions of "silence" before employing more computationally intensive comparisons.

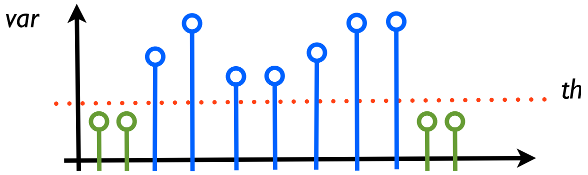


Figure 4: When finding start and stop points of a gesture or finding interesting regions in the EGL, we run a sliding window over the raw recorded time series and calculate the sample variance in that window when a new sample is inserted. If the variance is above a certain threshold, the gesture or interesting region starts. It stops when the variance falls below that threshold.

Searching the EGL parallelizes well, as each processor can be devoted to different regions of the EGL. However, even on a high-end, eight-core Macintosh workstation, searches were too slow for an interactive system. For a small, five-hour EGL with three-axis accelerometer data sampled at 40Hz, each example required between 5-25 seconds to check. Thus, one gesture with 10 examples could require minutes to search in the EGL. This slowness causes interface designers to create gestures in batch and then check them against the EGL. Testing a set of eight gestures with all their examples could take up to 20 minutes, leading to a relatively long and frustrating development cycle for the designer (Ashbrook and Starner, 2010). In the following sections, we describe a method to speed the EGL search using iSAX. We start with an overview of our method and our assumptions. We then provide the specific methods we used to adapt iSAX to our problem.

3.1 Overview Of EGL Search Method And Assumptions

In MAGIC Summoning, we first segment the EGL into interesting regions as defined previously. Each region is divided into four even subregions to form a “word” of length four. The region is then encoded into a string of symbols using the standard SAX quantization method. The string is entered into an iSAX tree representing the EGL. The iSAX tree is initialized with cardinality two but quickly grows as many regions hash to the same leaf on the suffix tree and the leaf needs to be split (Shieh and Keogh, 2008). As each region is encoded into the iSAX tree, its location in the EGL is recorded in the leaf. Once the EGL is completely encoded into an iSAX tree, we can perform “approximate search” using a gesture example as a query (Shieh and Keogh, 2008). The query is split into four regions and SAX-encoded in much the same way as the interesting regions of the EGL. An approximate search to determine the number of matches between the query and the EGL becomes a simple matter of matching the query string to the appropriate branch of the iSAX suffix tree and returning the number of strings contained in that branch.

One failing of this approach is that the interesting regions may be significantly larger or smaller than the candidate gestures. Regions significantly smaller than the command gestures are not of concern as they will never falsely match a command gesture in practice. We can eliminate such regions out-of-hand from the comparison. However, regions of movement that might match the query gesture may be hidden within longer regions in the EGL.

A key insight, which will be used repeatedly, is that we need not recover every region of the EGL that might cause a false match with the query. We are not intending iSAX to be used as a gesture recognizer. Instead, our goal is to allow the designer to compare the suitability of a gesture relative to other candidates quickly. As long as the movement occurs repeatedly in the EGL at isolated times as well as in longer regions, the iSAX method will report a number of "hits," which will be sufficient to warn the interaction designer of a problem.

A second insight is that users of gesture interfaces often pause before and after they perform a command gesture. Gesture recognizers exploit this behavior and use these pauses to help identify the command gesture. Movements that look like command gestures embedded in long regions of user motion are unlikely to be matched in practice by these recognizers. However, short everyday user motions that are similar to a command gesture are a particular worry for false positives. Thus, the iSAX encoding scheme of the EGL above seems suitable for our needs. However, if the goal of the interaction designer is to create gestures that can be chained together to issue a series of commands quickly, these longer regions in the EGL will need to be encoded more formally using constraints on how long a section can be encoded in each symbol. Such constraints can be derived from the length of expected command gestures (usually between 1-4 seconds in our experience), and the length of SAX word defined by the system.

A final insight is that a more precise comparison against the EGL can be made at the end of the gesture design process with the gesture recognizer that is output by MAGIC. During gesture design, all we require of the EGL search method is that it is fast enough to be interactive and that it provides an early warning when a given gesture may be susceptible to false triggering. Given the above operational scenario, we tune our iSAX implementation to provide fast feedback to the user. Details on the implementation follow below.

3.2 SAX Encoding

SAX quantizes time series in both time and value and encodes them into a string of symbols (Lin et al., 2007). For example, the time series in Figure 5 is divided into four equal portions (for a "word" length of four) and converted into a string using a four symbol vocabulary (a "cardinality" of four).

To be more precise, we first normalize the time series to have a zero mean and standard deviation of one. Assuming the original time series $T = t_1,...,t_j,....t_n$ has n samples, we want to first quantize the time series into a shorter time series $\bar{T} = \bar{t}_1,...,\bar{t}_i,...\bar{t}_w$ of word length w. The i^{th} element of $\bar{T}$ can be calculated by

$$\bar{t}_i = \frac{w}{n} \sum_{k=(\frac{n}{w}(i-1)+1)}^{\frac{n}{w}i} t_k.$$

Given the values in the compressed time series, we next convert them into symbols using a small alphabet of size (cardinality) a. Imagine the y-axis divided into an arbitrary number of regions bounded by $a-1$ breakpoints. Each of these regions is assigned to a symbol from the alphabet. Since we wish each symbol in the vocabulary to be used approximately the same amount, we place a normal Gaussian curve centered at 0 on the y-axis and place the breakpoints such that the area under the Gaussian for each section is equal. By performing the SAX process on the EGL and each gesture example separately, we are able to compare the changes in the signals through time without concern regarding their offsets from zero or relative amplitudes.

One convenience of the SAX representation is that there exists a distance calculation between two strings, defined as MINDIST by Lin et al. (2007), that is a lower bound on the Euclidean

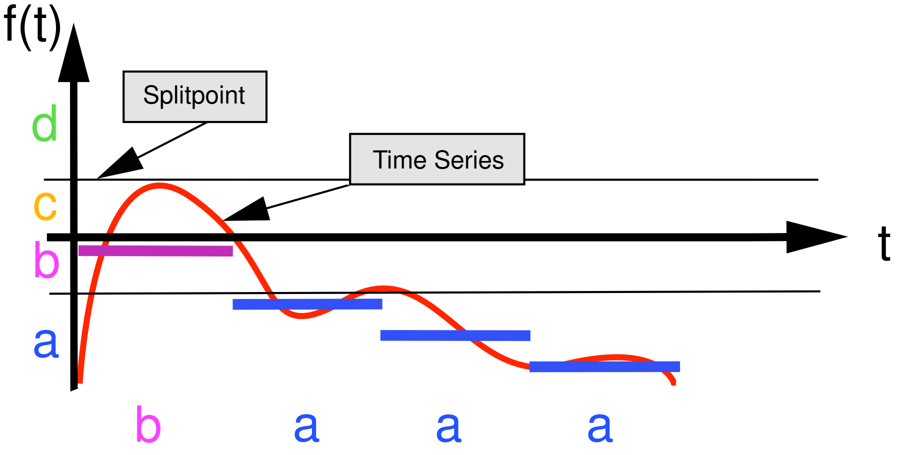


Figure 5: SAX process used to convert a time series to a string. The raw data is segmented into a user-specified word length, in this case four. Then each segment is replaced by a symbol associated with that region on the y-axis, based on the average value. The resulting string is represented by the string of symbols with superscripts indicating the number of symbols used to quantize each region: $b^4a^4a^4a^4$.

distance between the original two time series. Thus, we can search the EGL for possible false positives with some measure of confidence.

Another convenience of the representation is that the cardinality of each separate region can be increased whenever more precision is needed. For example, suppose we increase the cardinality of the first region in Figure 5 to eight (thus, the vocabulary would include letters a-h). The string might then be $d^8a^4a^4a^4$, as the region of the y-axis formerly covered by symbols a and b would now be covered by symbols a, b, c, and d. We can compare strings with regions of different cardinality by observing that we know that each time series is normalized before SAX encoding and that the regions are defined by a normal Gaussian centered at zero with all regions having an equal area under the Gaussian's curve. Thus, we still know the minimal distance possible between each region, and we can still use MINDIST to determine a lower bound on the Euclidean distance between the original two time series. This capability will be useful in our upcoming discussion on iSAX and its application to the EGL.

3.3 Multi-Dimensional iSAX Indexing And EGL Search

iSAX is a tree-based method for time series indexing introduced in Shieh and Keogh (2008). For encoding the EGL, our goal is create an iSAX tree that can be traversed quickly when searching for a match to a SAX-encoded example of a gesture. Each leaf of the tree contains the number of occurrences of that string in the EGL as well as the position of each occurrence. To begin, assume we are searching an EGL represented by the simple iSAX tree in Figure 6 with a query represented by $a^2a^2b^2b^2$ (for the sake of argument, assume we decided to represent the example gesture crudely, with regions of cardinality two). Immediately, we see that there is no branch of the tree with an a^2 in the first position, and we return no matches in the EGL. Now assume that we are searching the EGL for a query of $b^2b^2b^2b^2$. We find that there is a node of the EGL that contains that string, and that node has children (that is, the node is an "internal node"). Looking at the children in that branch, we see that we need to re-code the query gesture to have cardinality three in the first region. Re-coding reveals that the query gesture is better represented by the sequence $c^3b^2b^2b^2$, which matches one of the terminal leaves in the tree. The number of sequences from the EGL stored in that leaf is returned as the number of "hits" in the EGL.

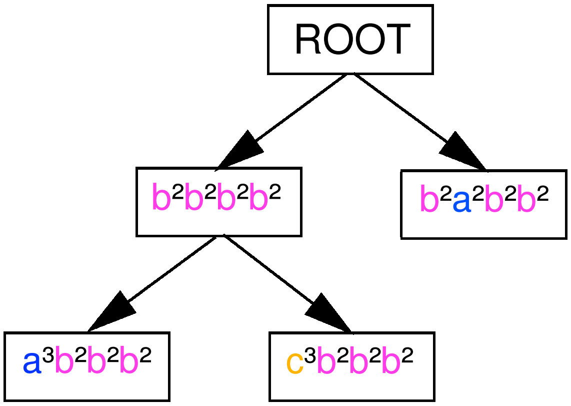


Figure 6: iSAX tree with three leaves. On the first level all symbols' cardinalities are equal. The node $b^2b^2b^2b^2$ is an internal node. For the children under this node, the cardinality of the first region is increased by one.

Next we describe how to encode a one-dimensional EGL into an iSAX tree. First, we find all the "interesting" regions in the EGL using the variance method discussed earlier. We divide the regions evenly into four sections and encode them using SAX with cardinality two, allowing for sixteen possible strings. Note that each node in an iSAX tree holds a hash table mapping child nodes to an iSAX word. Thus, when inserting a region into the iSAX tree, we compare the region's SAX string to the hash table in the root node. If there is no match, we create a child node and enter it into the hash table using its SAX string. If the SAX string is found, we examine the node to see if it is a terminal leaf. Each leaf points to a file (called a "bucket") stored on disk holding all of the regions that have mapped to it. The leaf also contains the position of each of the regions in the EGL and a count of the number of regions contained in the leaf. If the number of regions in the bucket exceeds a user specified size (called the "bucket size"), it is deleted, and the cardinality of the iSAX word is increased at one position (picked by round robin). At the deleted node's position we insert a new internal node. All the time series of the deleted node are inserted into the new node but with a higher cardinality. Children of the internal node are created as needed, effectively splitting the previous leaf into several new leaves. When we encounter a internal node during the insertion of a region, we search the node's hash table for children that match and proceed normally, creating a new leaf node if no matching child exists.

Note that this method of creating the iSAX tree dynamically adjusts the size of the vocabulary to better distinguish similar regions in the EGL. Given a bigger vocabulary, the SAX word will fit more exactly to the region. In other words, this method of encoding devotes more bits to describing similar movements that are repeated often in the EGL. Thus, when a query gesture is compared to the EGL iSAX tree, MAGIC will quickly return with no or few hits (depending on the specified bucket size) if the query is very distinct from the EGL. If the query is similar to motions in the EGL, the search process will traverse deeper in the tree, examining finer and finer distinctions between the query and the regions contained in the EGL.

The above discussion assumed that the data was one-dimensional. For multi-dimensional data, such as is used in the experiments described below, we create n iSAX trees, one for each dimension of the recorded data. We index all dimensions separately and join those n trees under one new root node (see Figure 7).

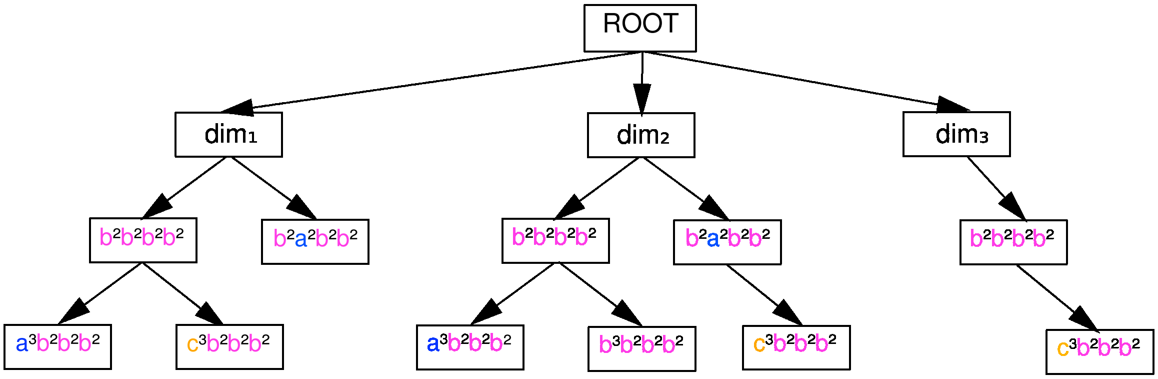


Figure 7: A multi-dimensional iSAX tree. Under the root node there is a dimension layer. Each node in this layer is the root node for a one-dimensional iSAX tree. During search, we search all iSAX trees, one for each dimension.

We query the EGL iSAX tree (constructed from the EGL) in all n dimensions. The result of that search is n files, one for each dimension. The number of hits can then be calculated by counting the number of places where each hit from each dimension overlap for all dimensions. Comparing the timestamps can be costly, so we introduced an approximation based on the observation that there can never be more overlapping time series than the number in the dimension with the lowest number of matches. For example, consider the result of a search in three dimensions (x, y, z) where the number of hits in the EGL are $x = 4$, $y = 20$ and $z = 6$. There can never be more then four hits total if we require that hits must overlap in all dimensions. The overall EGL testing method is summarized in Table 1.

Upon reflection, the EGL search procedure described above raises several questions and possibilities. What are reasonable values for the bucket size, word size, and cardinalities used in encoding the EGL, and how sensitive is MAGIC to these parameters? This question will be examined in detail in Section 6. A nice side effect of EGL search is that we can use the matches found to train a class of gestures that a recognizer should ignore (a "garbage" or NULL class). Section 5 will explore this option. Searching for which SAX strings are not contained in the EGL tree can suggest which gestures are not made during everyday movement. In Section 7, we exploit this attribute to recommend gestures to the interaction designer. However, first we will provide evidence that searching the EGL does indeed predict the number of false positives during the usage of a gesture interface.

```
Test preparation:
0) Collect a large data base of user movements in advance.
1) Find interesting regions by applying variance thresholding.
2) Build an n dimensional iSAX tree.

Gesture testing:
0) Find start and end point of gesture.
1) Search the iSAX tree in all n dimensions.
2) Return the number of time series in the minimum file.
```

Table 1: Testing gestures for potential false positives against a database of pre-recorded device usage.

4. Experimental Verification

In the following section we describe two experiments that suggest that an iSAX search of the EGL is a viable means to predict false positives. Our first goal is to show that false positive prediction using iSAX is correlated with the previous method of searching the EGL linearly using dynamic time warping (Ashbrook, 2009). We will also conduct an experiment in which we will show that the EGL is able to predict the relative number of false positives when using a gesture interface in everyday life. We describe the data used for the experiments and our experimental method before presenting our findings.

4.1 EGLs And Gestures Used In Evaluations

We use three different data sets to serve as EGL databases. The first is our Android accelerometer data set as described earlier. Before indexing the recorded data, we extracted the interesting regions, applying a threshold of $th = 0.001$ (triggering at almost any movement) and a window size of $N = 10$ (0.25 sec at 40Hz). The average duration of the interesting regions is $11,696ms$. The second EGL is based on the Alkan database[2] of everyday movements collected with an iPhone (Hattori et al., 2011). The third data set is another collection of everyday movements collected on Android phones for a different project at Georgia Tech. These two latter EGLs were processed in the same manner as the first.

We collected a reference data set of gestures for evaluation purposes. We acted as interaction designers and designed four gestures by performing them while holding a smart phone. For each gesture we collected 10 examples, resulting in 40 examples total. The four gestures are: drawing a circle in the air, touching your shoulder, shaking the phone up and down, and hacking (a motion similar to swinging an ax). The average duration of the gestures is between one and two seconds.

4.2 Comparison Conditions: NN-DTW And HMMs

When comparing the dynamic time warping EGL search method to a search in iSAX index space we will use the following procedure. The DTW method compares each interesting region from the EGL to each gesture example (Ashbrook, 2009). We calculate the dynamic time warping distance

[2] Alkan web site can be found at: `http://alkan.jp/`.

of a new gesture to all examples in the EGL and apply a threshold chosen empirically. All regions for which the distance is below this threshold for any example count as a false positive (in keeping with MAGIC's ability to output a one nearest neighbor classifier for live gesture recognition).

For yet another comparison, we use hidden Markov models to search the EGL for false positives. For the experiments in this paper, we use a six-state HMM (ignoring initial and end states) with one skip transition and one Gaussian output probability per state per dimension (see Figure 8). We collect all the examples for our gesture set first and then train a HMM for each of the gestures. We classify each region in the EGL and apply a threshold based on maximum likelihood to determine if a region in the EGL is close enough to the gesture to count as a false positive. We chose both the maximum likelihood threshold as well as the distance threshold so that classifier accuracy stayed high (93% for NN-DTW and 100% for HMM).

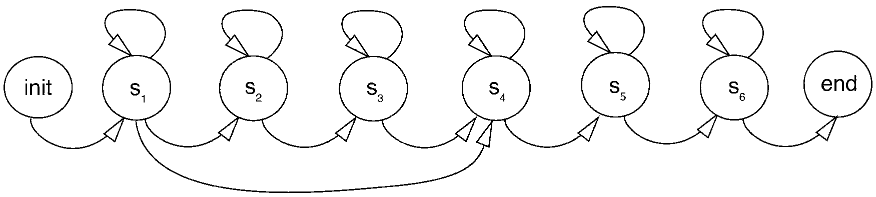


Figure 8: The topology of the left-right, six-state HMM used in our experiments. The first state is the start state, and the eighth state is the end state. Each internal state transitions to itself and its successor. We include a skip transition to help recognize shorter gestures.

4.3 Comparison Of iSAX To NN-DTW And HMM In Searching EGLs

We wish to compare our iSAX EGL search method to the more conventional NN-DTW and HMM techniques described above. When selecting between two candidate gestures, the interaction designer wishes to choose the one with a lower number of predicted false positives. Thus, if a first gesture has few hits when NN-DTW or HMMs are used and a second gesture has many hits, that same trend should be shown with iSAX. The absolute number of EGL hits does not matter, but there should be a strong correlation between the relative number of hits returned by iSAX and the other two techniques when run on the same set of gestures. We use the Pearson correlation coefficient as a metric to compare the techniques.

Regardless of the search method used, we store the number of hits in a vector. Each entry of that vector corresponds to the overall number of false positives for a given gesture. For iSAX and NN-DTW, the overall number of false positives for a gesture is calculated by searching the EGL for each example of that gesture and summing the resulting numbers of hits. For HMM models, thresholding on the log likelihood probability is used. For our set of four test gestures, testing returns three vectors (one for each method) of four elements (one for each gesture). We calculate the Pearson correlation coefficient between the iSAX vector and the NN-DTW vector and between the iSAX vector and the HMM vector.

To reassure ourselves that this technique produces a meaningful metric, we performed Monte Carlo simulation experiments. Indeed, the correlation of random vectors with four elements show low r values.

First, we compare the search methods on the EGL from Bremen. We chose the iSAX parameters empirically:

word length: 4

base cardinality: 2

bucket: 6000.

Figure 9 compares the number of hits per hour returned by each method. The hits per hour metric reflects the number of matches found in the EGL divided by the original time required to record the EGL. One can see that our iSAX search approximation returns many fewer hits than NN-DTW or HMMs. However, the magnitude of the iSAX values correlate strongly with the NN-DTW ($r = 0.96$) and HMM ($r = 0.97$) results. Thus, a high number of hits returned by iSAX on the EGL (high compared to other gestures tested with iSAX) is a good indicator for when a gesture should be discarded. The remaining gestures are suitable candidates for user testing.

We also measured the time needed to complete the search for each method on a 2.0GHz Intel Core Duo T2500 Macbook with 2GB of RAM. The NN-DTW and HMM methods require more then 10 minutes to complete the search on all 40 gesture examples whereas iSAX search required 22 seconds, a 27X increase in speed. With such speed, each of the gesture examples could have been checked as it was entered by the interaction designer. In fact, the EGL search would require less than a second for each gesture example, which is less than the amount of time required to check a new example for confusion against all the other gesture examples with NN-DTW when creating a eight gesture interface (Ashbrook, 2009). Thus, we have obtained our goal of maintaining interactivity during gesture design.

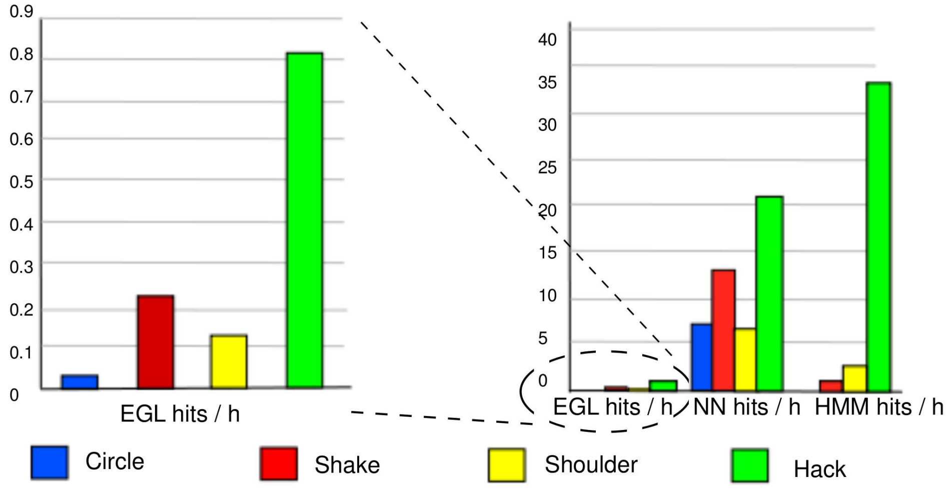


Figure 9: *Left:* The hits per hour in the EGL based on iSAX search. *Right:* A comparison of the number of hits per hour returned by iSAX, NN-DTW, and HMMs from the EGL.

We were curious as to how much EGL data is needed to predict poor command gestures. We generated three random subsets of the EGL by picking 100, 200 and 500 interesting regions at random from the data set and comparing the correlation coefficient between iSAX and NN-DTW. The correlation between the results remained surprisingly high, even with an EGL containing only 100 regions:

- **n = 100:** $r = 0.89$
- **n = 200:** $r = 0.93$
- **n = 500:** $r = 0.93$.

As later experiments show, more data is better, but even a relatively small EGL can help the interaction designer avoid choosing troublesome gestures. We also compared iSAX versus NN-DTW in the Alkan and Georgia Tech EGLs, with similar results to the original Bremen EGL:

- **Alkan:** $r = 0.94$
- **Georgia Tech:** $r = 0.99$.

Our results suggest that the results of an iSAX search on the EGL correlate highly with those of the slower EGL search methods. Even though the absolute number of hits found by the iSAX method are significantly fewer than the other methods, the relative number of hits can be used to compare the desirability of one candidate gesture versus another.

4.4 Comparison Of iSAX Predictions To HMM And NN-DTW Gesture Recognizer Use In Practice

Next, we examine whether our iSAX EGL search method is able to predict false positives in everyday life. In fact, this experiment is the first to verify that any EGL search is able to predict false positive rates of a gesture recognizer in practice.

We exported NN-DTW and HMM recognizers from MAGIC Summoning for the four gestures trained during the process described in the previous experiment. We integrated the HMM classifier into an interactive system. Next, we recruited four Android phone users who had not contributed to the EGLs nor the training of the gestures.

In order to understand how difficult it was to perform the gestures correctly, we asked the users to perform each gesture 10 times without feedback. The HMM classifier performed at 60% accuracy, which is not surprising given the gestures and testing procedure. Next we allowed the users to train with the HMM recognizer to become more familiar with how to perform the gestures so that they could be more easily recognized. This way of learning can be found in commercial systems like the Nintendo Wii, which uses avatars to help users learn control gestures. Not surprisingly, the four users' average accuracy with the HMM recognizer improved to 95% after training.

After the users completed their training, we installed a software application on their phones that notified the users when to perform one randomly selected gesture, once every hour. Otherwise, the users performed their normal activities, and the application records all the users' movements. We searched the recorded data for the intended gestures. The HMM classifier found $50\% - 70\%$ of the intentional gestures whereas NN-DTW search found all of them. However, the NN-DTW classifier

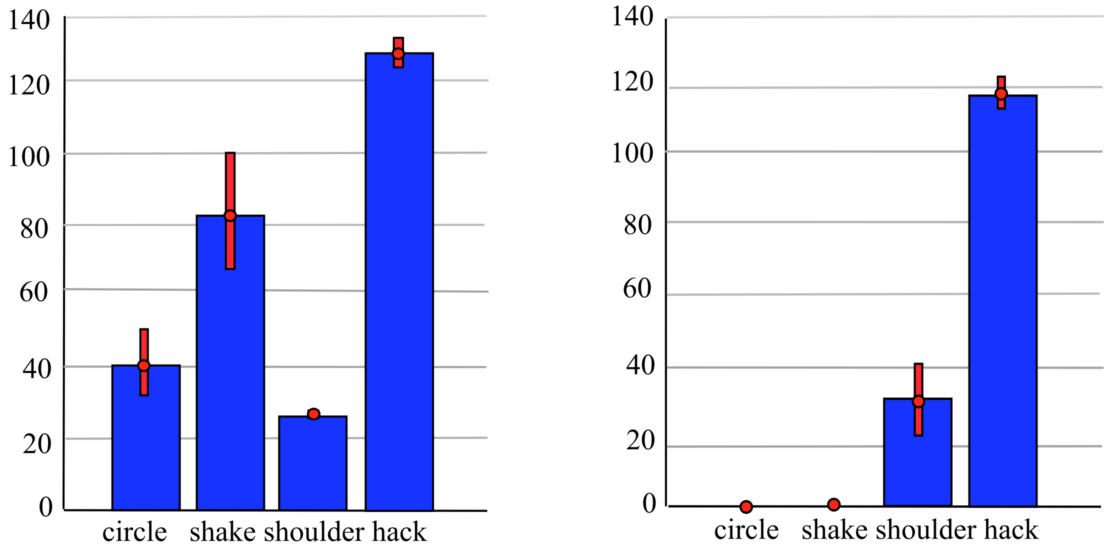


Figure 10: The EGL hits per hour found during deployment. *Left:* The EGL hits for NN-DTW search per gesture. *Right:* The EGL hits for HMM search per gesture. The EGL hits for a gesture are the average hits over all four users. The bars correspond to one standard deviation.

had lower precision than the HMMs. Given that we specifically allowed gestures that were known to be poor (from EGL testing) and that the system did not provide feedback to the users, such poor performance is to be expected (and desired from the point of the experiment).

Figure 10 shows the false positive rates for each gesture and recognizer. We observed a high correlation ($r = 0.84$) between the relative false positive rates predicted by the iSAX search on the original EGL and the actual, tested NN-DTW performance on the users' data. The correlation was even higher ($r = 0.97$) for the HMM classifier. These results support our hypothesis that MAGIC Summoning can be used to predict gestures at risk of having many false positives when deployed in gesture recognizers in practice.

5. Improving Recognition Through A NULL Class Created From EGL Search

In the experiments in the previous section, we needed to specify a threshold to avoid false positives when distinguishing the four gestures from our four users' everyday motions. For NN-DTW, the threshold was a distance, while with HMMs it was a probability. Setting this threshold requires more pattern recognition experience than an interaction designer may possess, and often gestures are not separable from everyday movements with a simple threshold. Another option is to create a NULL (garbage) class, which attempts to capture all the motion not matching the gestures of interest. With this technique, the recognizer runs continually but does not return a result when the sensor data matches the NULL class.

Here, we use EGL data to train a NULL class automatically so that a user-defined threshold is not needed. Multi-dimensional iSAX search of the EGL returns time series similar to a query gesture. Thus, it is a simple matter to collect the EGL hits from all examples of all gestures in the gesture interface to train a NULL gesture (using either technique).

The following experiment is based on the data collected while our four users performed the four requested gestures during their daily activities. We adjusted the thresholds upward for the HMM and NN-DTW recognizers to avoid misclassifications in the EGL while still detecting the gestures from the training set. We also trained NULL classes for both recognizers. Figure 11 shows the results of all four recognizers running on the user study data. Using the EGL NULL class method resulted in a statistically significant improvement of both the NN-DTW ($p << 0.0001$) and HMM ($p < 0.05$) recognizers. Both avoided more false positives using the NULL class instead of a threshold. Gesture recognition accuracy and correlation to the iSAX EGL hits remained consistent with the experiment in the previous section. The results suggest that training a NULL class based on EGL hits can be a successful way to improve performance and reduce complexity for the interaction designer. Note that many variations of this technique are possible and might further improve results. For example, a different NULL class could be trained for each gesture.

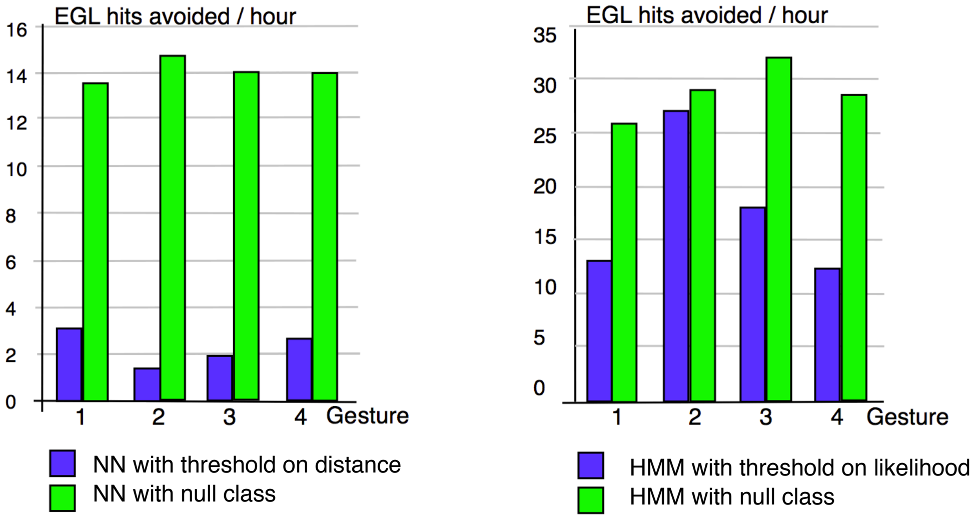


Figure 11: *Left:* The false positives per hour avoided using a NULL class for each gesture based on EGL hits versus the simple threshold. *Right:* The false positives per hour avoided for HMMs using the NULL class versus the simple threshold.

6. iSAX Parameter Sensitivity Experiments

In Section 4.4, iSAX was able to predict the relative performance of a gesture during continuous recognition. However, the process required setting several parameters: word length, bucket size, and initial cardinality. In addition, we compared the false positive predictions to that of the NN-DTW method, which itself required a distance threshold (when a NULL class is not used). How sensitive is our method to these parameters? We use the same four test gestures (circle, shake, shoulder, hack) and EGL as in our user study to explore this issue.

Observe that the cardinality of the sequences is automatically adjusted during the creation of the EGL iSAX tree, quickly changing from its initial minimal setting of two. Effectively, this parameter is not set by the user, and we can remove it from the following experiments on parameter sensitivity by holding it at a reasonable value. We choose a base cardinality of four ($card = 4$), given that this level of complexity was judged sufficient from observations in the original iSAX experiments (Shieh, 2010; Shieh and Keogh, 2008) and in our own work (Kohlsdorf et al., 2011).

In the experiments below, we compare the iSAX results, while varying the bucket size and word length, to the NN-DTW method using the correlation method described above. We also tried comparing the iSAX method to NN-DTW with different reasonable distance thresholds (3.3, 5, 10), but we found little change in the results. For example, the bottom of Figure 12 shows graphs comparing iSAX word length to correlation with NN-DTW at each of the distance thresholds. The graphs are very similar, indicating that the comparison with NN-DTW is relatively stable with respect to the distance threshold used. Thus, we turn our attention to word length and bucket size.

In the first part of the experiment we test the correlation of the EGL searches using NN-DTW and iSAX trees constructed with different iSAX word lengths (4,5,6, ... ,13) and bucket sizes (1000, 2000, ... , 10000). Figure 12 plots the results. Changes in bucket size cause minor variations in correlation; however, word length has significant effects.

Since the performance of our method seems mostly dependent on one parameter, we propose an automatic parameter tuning method that does not require any data except a pre-recorded EGL. The central concept is to choose random regions from the EGL to serve as a gesture training set and to tune the iSAX parameters to that set using hill climbing.

We require the user to specify the number of gestures in the data set (N), how many examples we want to collect for each gesture (M), and a threshold on the dynamic time warping distance over which two time series are distinct. We pick N regions of motion (“interesting” regions) at random from the EGL to serve as “reference gestures.” For those N reference gestures we extract M examples from the EGL where the DTW distance to the reference gesture is smaller than a threshold. Then we compute the false positives for this gesture set using the NN-DTW method. In order to find the appropriate word length we use hill climbing in the iSAX parameter space. At each step, we perform false positive prediction using iSAX and compare the results to the NN-DTW results using the Pearson correlation coefficient as an objective function.

We ran an experiment to test this hill-climbing technique, allowing the procedure to set the word length automatically and comparing the results to NN-DTW. We started the word length at 4 and increased it to 13. If the observed correlation at a given word length is followed by a smaller one when the next word length is tried, the algorithm stops and returns the last word length. As one can see in Figure 12, after 3 iterations iSAX finds a local maximum. However, this sequential method is not optimal. For example, if the word length which maximizes the correlation is 9 and the local maximum at the word length 6 is smaller, we would stop too early. However, this problem can be solved by including simulated annealing or stochastic gradient descent in the future.

In this chapter, we showed that the iSAX EGL search relies on several parameters but that the parameters can be tuned automatically. Word length seems the primary parameter that needs to be tuned.

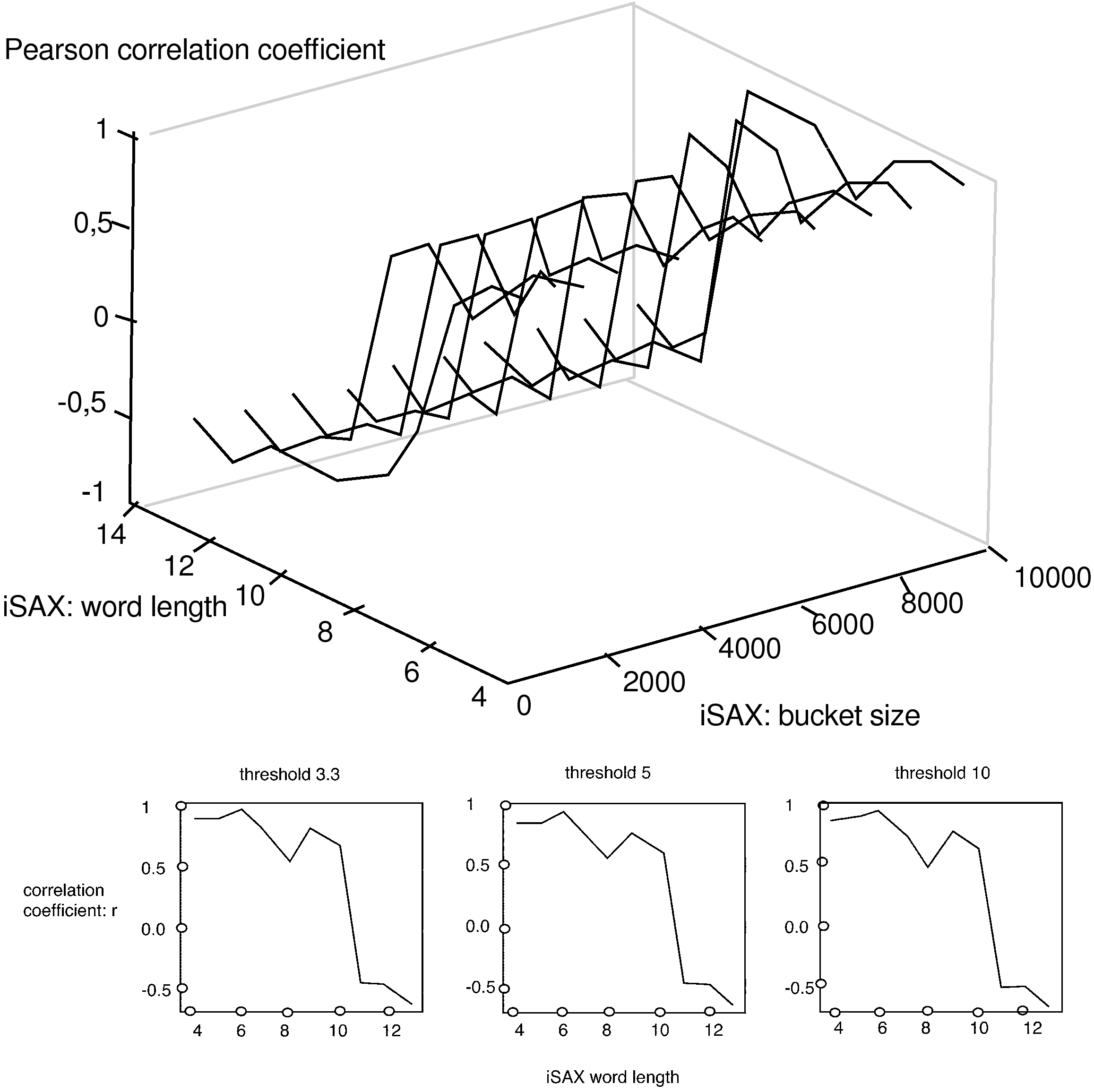


Figure 12: Top: correlation to NN-DTW vs. iSAX word length vs iSAX bucket size. Bottom: iSAX word length vs. correlation to NN-DTW for distance thresholds of 3.3, 5, and 10, respectively.

7. MAGIC Summoning: Suggesting Gestures With Low Probability Of False Positives During Use

To this point, we have focused on efficient gesture testing. However, when using MAGIC to design gestures in previous studies, our participants wished to have MAGIC suggest potential gestures instead of creating their own. Often the gestures designed by the participants showed high false positive rates when tested against the EGL, leading to frustration. MAGIC users said they would rather select from a set of gestures that were “known good” than experiment blindly with constraints they did not understand (Ashbrook, 2009).

In the next section, we describe a method for suggesting gestures based on a pre-recorded EGL. We then perform an experiment where we test suggested gestures for false positives during normal device usage by naive subjects. Finally, we examine different possible metrics to order the suggestions for easier selection by the designer. While we have mostly used accelerometers in our experiments to date, here we concentrate on capacitive trackpads, specifically those used on Apple’s laptops. Data from inertial sensors are hard to visualize for an interaction designer without a inverse kinematic system to map the sensor readings into limb movement. While such systems are now feasible with adequate accuracy, we wished to avoid the additional complexity for these first experiments. Trackpads provide two dimensional data that are easy to visualize for an interaction designer, and trackpads are commonly used in everyday office work. In addition, industry has begun to include more complex command gestures in their trackpad-based products (Li, 2010).

7.1 Synthesizing And Visualizing Gestures

We introduce a method for proposing gestures that do not collide with every day movements using four steps, briefly outlined here. First, we collect an EGL that is representative of the usage of the device or sensor. Next, we build an iSAX tree based on the EGL. We systematically enumerate the possible SAX strings and check for those which are NOT contained in the tree. Finally, we visualize these gestures and present them to the interaction designer. Once the designer selects a set of gestures for his interface, MAGIC Summoning can train a recognizer for the gestures using synthesized data.

7.1.1 COLLECTING AN EGL

Collecting a representative EGL is often time-consuming and is best done by someone familiar both with the specific sensor involved and pattern recognition in general. Fortunately, the process is only necessary once for the device of interest and then can be used for different interface designers and tasks. Mostly, the EGL will be collected across multiple people to ensure that the resulting gestures can be user independent. Ideally, the EGL should be collected across every situation and physical context where the device might be used (for example, sitting at a desk or driving) to make sure that incidental motions are well represented. If the resulting gesture recognizer is intended to work across different devices (for example, across multiple version of Android phones), the EGL should be collected from a representative sample of those devices.

7.1.2 REPRESENTING THE EGL AND GENERATING GESTURES

Next, we convert the EGL into a simplified iSAX tree structure. Unlike the work above, here we only care that a given string occurred in the EGL instead of how many times it occurred. Thus, we can use a simpler indexing method that will allow easier gesture building later. We convert interesting regions from the EGL to SAX words and build the set of all strings observed in the EGL. Since the sensor input is multivariate, we build the SAX word in each dimension and concatenate the words. Thus, for n dimensions and a word length of w, the indexing key grows to $n * w$. Given the cardinalities in the word, discovering gestures that are not represented in the EGL is a simple matter of combinatorics. We generate all possible gestures and store the gesture as a viable candidate if it is not contained in the EGL.

7.1.3 VISUALIZING CANDIDATE GESTURES AND TRAINING GESTURE RECOGNIZERS

In order for the interface designer to select between the different candidate gestures, we must visualize them. Specifically, we need to convert the candidate gesture from a SAX string into a real valued time series. For each SAX symbol, we know that valid values are somewhere between the upper and lower breakpoint of the area assigned to the symbol. We choose a random point between these breakpoints for each symbol. We then use spline interpolation or re-sampling to fit a curve through the resulting values from each SAX symbol. We used an exponential moving average to smooth the resulting curve. The overall process is shown in Figure 13. Note that by repeating this process we can generate a synthetic set of time series that could have generated the SAX word. This synthetic data is used to visualize acceptable versions of the trackpad gesture to the interaction designer. We will also use this synthetic data to train a recognizer for the gesture if it is selected (see below).

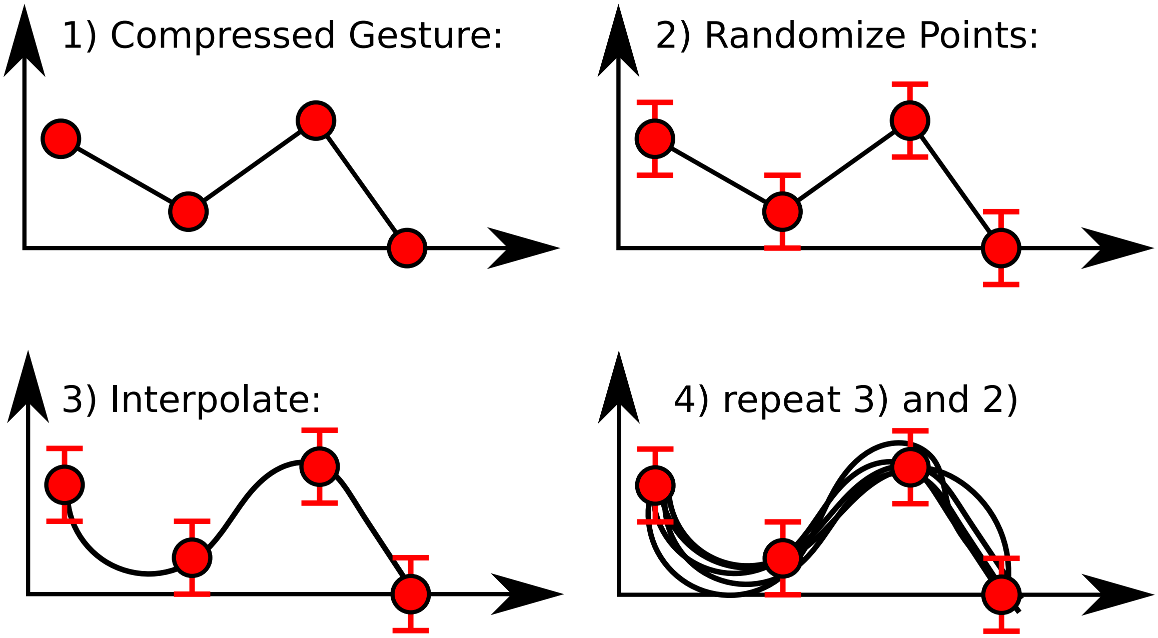


Figure 13: Converting a SAX word to example gestures.

Figure 14 shows MAGIC Summoning's user interface for gesture suggestion. In the center of the window we display a synthesized gesture. The color of the lines indicates the time course of the gesture as it is performed on the trackpad (from dark to light). Many synthetic examples of a given SAX word are drawn to give the interaction designer a sense of the possible shapes of the gesture. New suggestions are displayed periodically, effectively creating a movie of potential gestures. In our first implementation, gesture suggestions were selected randomly, keeping a list of previously viewed gestures so as to avoid repetition. If the interaction designer sees a desirable gesture, he stops the presentation with a key press.

If other gestures have already been selected by the user, the similarity of the currently displayed gesture to the already selected gestures is shown in a bar plot in a window at the bottom left. Based on these similarity scores, the user can retain the gesture or continue searching other suggestions. In this case, we decided to use the $1 Recognizer (Wobbrock et al., 2007) both for generating

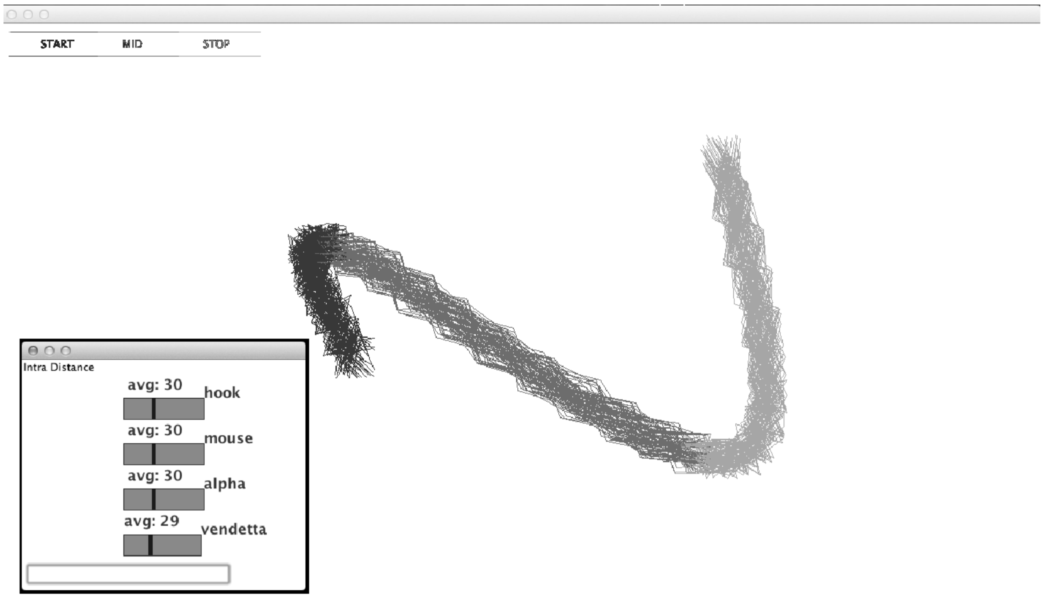


Figure 14: The MAGIC Summoning gesture suggestion interface.

similarity scores and for gesture recognition. To train the gesture recognizer, we simply used the synthetic examples generated during the visualization process.

7.1.4 $1 RECOGNIZER

Since the $1 Recognizer is widely used in HCI research (Belatar and Coldefy, 2010; Dang and André, 2010) but is not necessarily known to machine learning researchers, we give a quick overview here. The recognizer is optimized for single stroke gestures and can be considered instance-based learning. Each instance or template is re-sampled to be of equal length with all others and then rotated, scaled, and translated to a canonical form before being used. During recognition the query gesture is compared to all the stored templates using an angular distance metric. In continuous recognition we can apply a threshold on that distance, and the rest of the recognition process is similar to the dynamic time warping approach described earlier. The authors report recognition accuracies of 99%, which is comparable to DTW implementations on the same data sets. The method is simple, fast to compute, and understandable by pattern recognition novices. Thus, the algorithm is well-suited for experimentation by interface designers. With MAGIC Summoning, interaction designers do not need to collect any training data for the recognizer. The training data is produced synthetically from the EGL as described above. Note that we can use the $1 Recognizer as a distance measure for EGL search (albeit slowly compared to iSAX), which will be useful for comparison experiments below.

7.2 Testing Suggested Gestures And Recognizers In Practice

We collected an EGL consisting of ten participants using their personal Mac laptops for one week. Figure 15 visualizes the EGL. While indexing the EGL, we set the SAX word length to four. For a two dimensional touchpad, the length doubles to eight. Setting the cardinality to four leads to a total number of 65536 (4^8) possible strings.

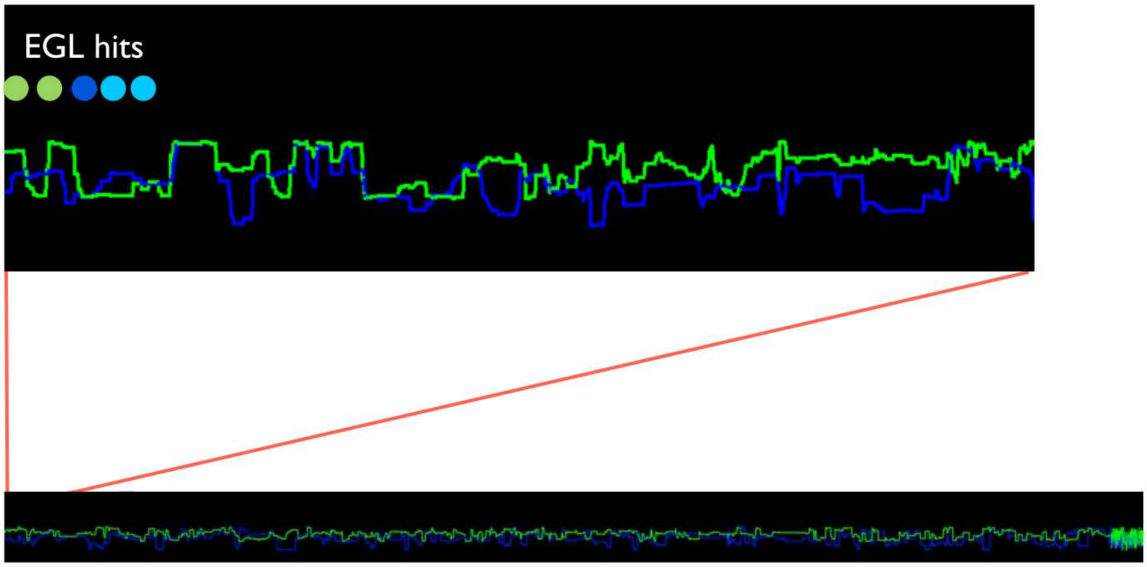


Figure 15: Bottom: The touch pad EGL. Top: An excerpt from the EGL showing five false positives during testing of a gesture, indicated as colored bubbles.

We observed 1222 unique strings in the collected EGL. The space is surprisingly sparse; there are 64314 strings not found in the EGL, suggesting that there are a large number of gestures that could be made with a low probability of false positives.

We performed an experiment to evaluate if the proposed suggestion and selection process described in the previous section can produce gestures that show a low false positive rate in everyday life. In addition, we were concerned as to whether synthetic data would be sufficient to train a high accuracy recognizer for this domain. We acted as an interaction designer and selected six gestures using the visualization tool above (see Figure 16). We preferred gestures that were simple and memorable. Figure 17 demonstrates 70 other gestures suggested by the system that were not used. We trained a $1 Recognizer for each of the six gestures selected using synthetic data generated by MAGIC.

We designed a six user study with users who did not contribute to the EGL. As in the false positive prediction experiments from the previous section, we asked users to practice with the recognition system so that they could perform the gestures with confidence. Users were able to improve their performance from $\approx 46\%$ to $\approx 90\%$ quickly. Afterward, the users worked on their computers for four hours while all touchpad movements were recorded. Every 10 minutes we sent a notification to the users asking them to perform one of the six gestures, resulting in four examples of each gesture for each participant. Thus, we collected 24 hours of data and 144 gesture examples.

The gesture recognizer was able to recognize 98% of the performed gestures. Even though synthetic data was use to train the recognizer, these findings are similar to those of Wobbrock et al. (2007), who reported a 99% accuracy in their experiments. The false positive rates of the gestures

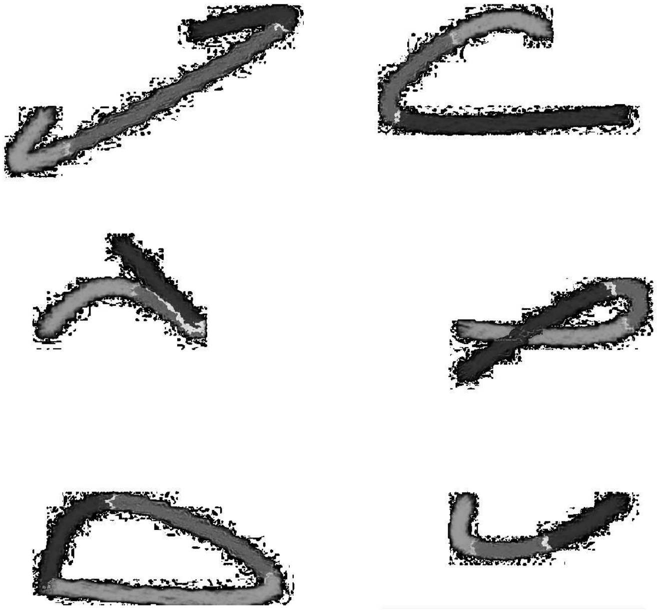


Figure 16: The six gestures used in the study. Gestures are drawn from dark to light.

are low except for one gesture (see Figure 18). Thus, the experiment supports the hypothesis that MAGIC Summoning can suggest gestures and aid the interaction designer in creating a gesture system that results in low false positives. However, several questions remain. Can we order the suggestions so as to present the "best" gestures first? Also, the experiment as described has no control condition. What would have been the result if we had tried suggesting random gestures from the $64,314$ available?

7.3 Ordering Gesture Suggestions

In this section we will explore possible ways of ordering gestures such that users can quickly find desirable gestures from the large number of possibilities. Our intuition is that users prefer simple gestures since they can be accessed quickly and are easy to memorize.

Our first approach is defining the complexity of a gesture as the entropy of its SAX word (Mitchell, 1997):

$$H(word) = -\sum_{i=0}^{card} p(symbol_i) * log(symbol_i).$$

However, if we want to prefer simpler gestures, we should check to determine if false positive rates in real usage are correlated with simplicity. Otherwise, proposing simpler gestures first could be counterproductive. Intuitively, one would think that simpler gestures would trigger more often in everyday life. To investigate this question we trained the \$1 Recognizer with 100 randomly chosen gestures and searched the EGL with it. For each gesture we calculated the entropy and compared the false positive rate to the entropy and found no correlation ($r^2 \approx 0.04$). Thus, there seems to be little additional risk to suggesting lower entropy gestures first.

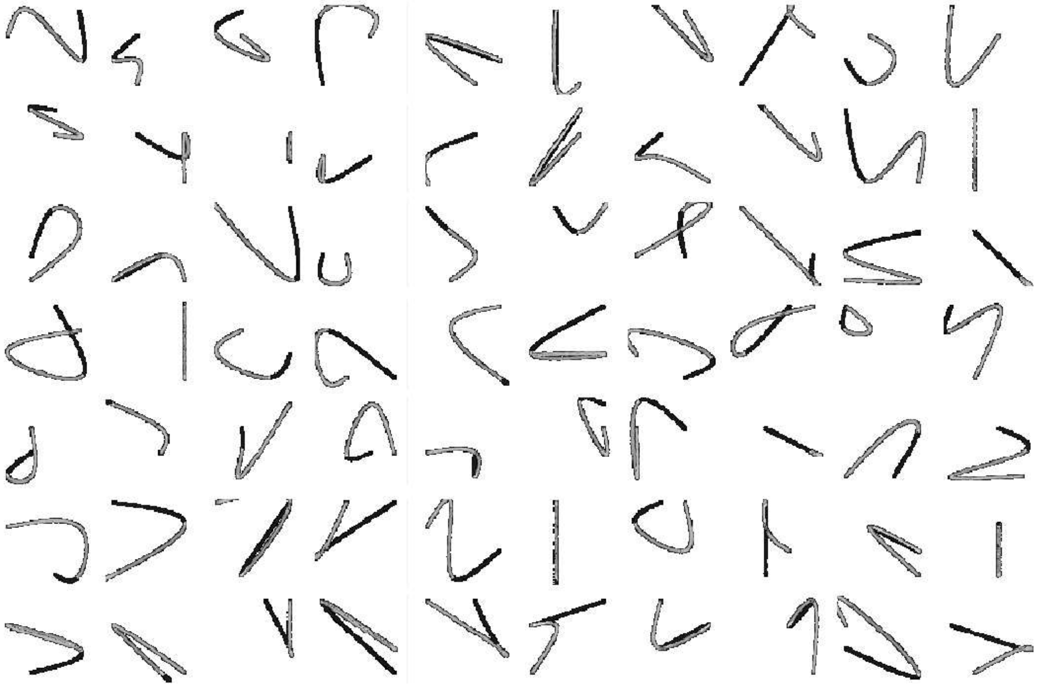


Figure 17: 70 generated gestures with potential low false positive rates. Gestures ordered from left-to-right and from top-to-bottom with increasing entropy.

The above heuristic seems logical for ordering suggestions. Low entropy gestures would seem to be simpler and easier to perform. To confirm this intuition we ran a small user study. We generated 100 gestures and sorted them using the above score. We examined the 20 best-ranked gestures and rejected ones that required significant overlap of the strokes (see Figure 19) as the static visualization of the strokes could confuse subjects. For each of the 10 remaining gestures we asked six users to perform the gesture in the air, on the table or on their touchpad and asked them to assign a score of performability between 1 and 10. All participants received the same gestures. Interestingly, we were not able to find a correlation between the entropy of a gesture's SAX word and the users' ratings ($r^2 = 0.09$).

Given the above result, we desire gestures not in the EGL but that are known to be performable. With a trackpad, all suggested gestures should be physically possible, but in future work with inertial sensors the suggestions could become impossible without constraining the system in some manner.

We decided to prefer gesture suggestions where the substrings of the SAX word representing the candidate gesture are represented in the EGL, but the gesture string itself was not present. We will assume one dimension for ease of illustration. If a gesture ACBD is not in the EGL, but the subcomponents AC, CB, and BD or ACB and CBD were well represented in the EGL, we might conclude that ACBD is possible for the user to perform. In other words, we will prefer gestures where the most n-grams from the EGL are included in the suggested gesture's string. Intuitively, though, such a heuristic causes concern that such gestures might have a higher chance of false triggering.

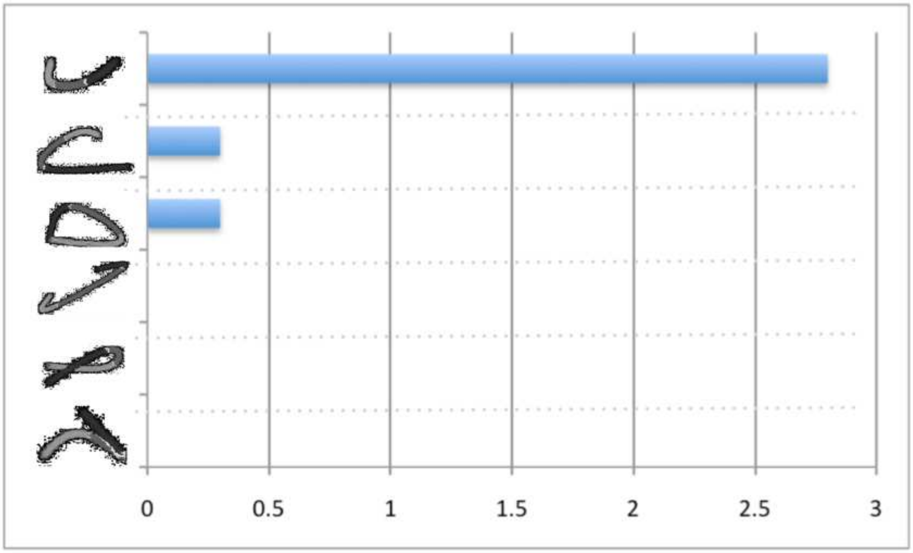


Figure 18: Results of the trackpad gesture user study in false positives per hour. All but one of the gestures suggested by MAGIC Summoning show a low false positive rate.

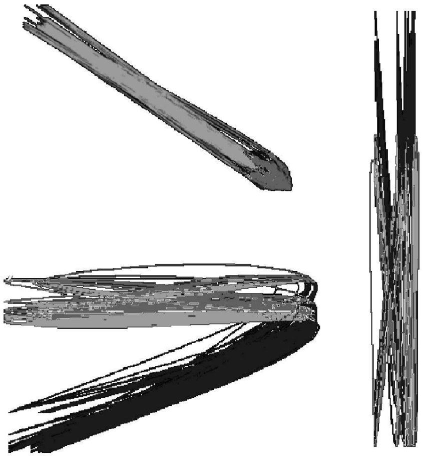


Figure 19: MAGIC Summoning gestures with significant overlap of the strokes were rejected to avoid user confusion.

To investigate this possibility, we extracted bi-grams and tri-grams from the EGL, created candidate gestures from them, and tried to find a correlation between the false positives in the EGL and the number of n-grams in the gesture's string. Note that this method of composition creates gestures with a variety of properties: ones common in the EGL, rare in the EGL, and not present in the EGL.

A correlation would indicate an increased risk with this method of ordering the suggestions, but we did not find one, giving a modicum of assurance in the method:

Bi-grams $r^2 = 0.000676$

Tri-grams $r^2 = 0.000256$.

Beside low false positives, another criteria for a good gesture system is that there should be a low chance of confusion between gestures. If the user is creating a control system with six gestures and has already selected five of them, we should prefer suggestions that are distinct from the five gestures already chosen. We measure the distinguishably of a gesture using the Hamming distance (Hamming, 1950) of the gesture's SAX word. Thus, when ordering gestures, we sort using a score defined as

$$score(word) = \frac{dist(word)}{(1 + entropy(word))}$$

where the distance of the word is the average Hamming distance to all other gestures in the gesture set. This metric provides a high distance to the other gestures and a low entropy. Note that we use $(1 + entropy(word))$ to avoid unreasonably high or infinite scores when the entropy value is near 0.

Given the results of the above experiments, we are now tuning MAGIC Summoning to generate gestures composed from parts of the EGL and to suggest gestures that are most dissimilar to each other. We intend to test this ordering system in our future work with suggesting gestures for use with inertial sensors.

7.4 How Selective Are MAGIC Summoning's Suggestions?

In the above user study, we selected six gestures by hand from MAGIC Summoning's suggestions and tested the $1 Recognizer that MAGIC output for both accuracy and false triggering. However, there were many possible gestures that the system could have output instead. In this last section we will investigate if suggesting gestures based on our method is better generated ones by chance.

As we have seen previously, using iSAX results in fewer hits being identified in an EGL than those found by typical gesture recognizers (HMM, NN-DTW, $1 Recognizer, etc.). The sole reason to use iSAX is that it quickly returns whether or not a candidate gesture is worthwhile to investigate further. However, we do not need to generate gesture suggestions in real time. In fact, as soon as an EGL is collected, the same "overnight" process that generates the EGL's iSAX tree representation for prediction could track the gestures not represented in the EGL. Once these gestures are known, the recognizer of choice could be trained with synthetic data of the gesture, and the recognizer could be run on the EGL for a more precise estimate of the expected hits. The number of false positives returned should allow a finer discrimination between candidate gestures. In the following experiment, we use this new procedure to generate suggested gestures and test ones with the lowest number of false positives on the test data collected from subjects not represented in the EGL.

In this experiment, we generated 2000 random gestures from SAX strings not in the EGL. For each of the gestures we synthesized 40 examples and trained a $1 recognizer with them. We used this recognizer to test search the EGL in the classic way, that is testing each interesting region using the trained recognizer. We used a typical threshold ($th = .85$) for the $1 score. All results above that threshold count as a hit with the EGL. Figure 20 orders the gestures by least to most number of hits per hour in the EGL. Clearly the $1 Recognizer identifies many potential false positives, yet most of the gestures still have low rates.

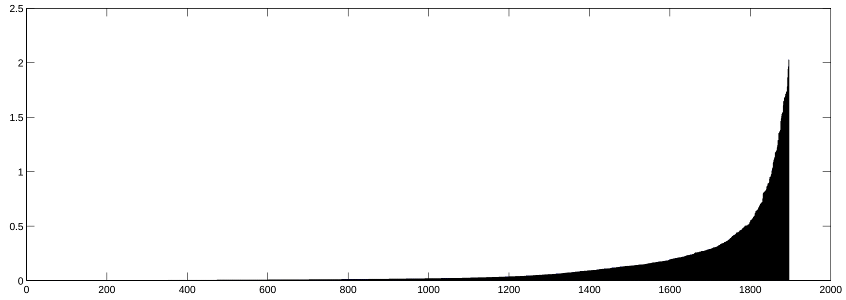


Figure 20: Number of false positives identified in the EGL using the $1 Recognizer for each of 2000 gestures synthesized from SAX strings not represented in the EGL. Gestures with more than 2 hits per hour are not graphed to preserve scale.

Figure 21, top, shows another view of this data. Note that over 35% of the 2000 gestures have 0 - 0.0001 false positives/hour. Compare this rate to that of Figure 21, bottom. This graph was generated using all the SAX strings represented in the EGL. Less than 12% of these gestures have such low false positive rates. Clearly, the SAX representation does have considerable predictive power on which suggested gestures are least likely to trigger falsely using the $1 Recognizer in the EGL. In fact, better than one in three of the gestures suggested by choosing SAX strings not in the EGL will be candidates for very low false positive rates with the synthetically trained $1 Recognizer.

The above observation suggests a relatively efficient method for creating gesture candidates for the interaction designer. First, randomly choose a unrepresented SAX string in the EGL. Train the desired recognizer using synthetic data. Run the recognizer on the EGL. If the rate of false positives per hour is less than 0.0001, keep the gesture. Otherwise, discard it. Generate as many gesture suggestions as is possible given time constraints. (Approximately 25 minutes is required to generate 100 gesture suggestions using a modern laptop, but such a process is highly parallelizable and can be run in batch before the interaction designer approaches the system.) Order the suggestions as described above and present them to the interaction designer for selection.

We conducted an experiment evaluating this algorithm. We split the collected EGL for touchpad gestures into two subsets. Each subset contains randomly chosen, distinct time series from the original EGL. The intersection between the subsets is empty. We used the first subset to generate 100 randomly chosen, distinct gestures candidates that show less then 0.0001 false positives per hour using the $1 Recognizer. We used these recognizers to then search the data in the second subset. On average we found the gestures to trigger 0.0022 times per hour, with a standard deviation of 0.003. These rates correspond to an average time between false triggerings of 455 hours, or approximately one month assuming usage 16 hours/day. Thus, this method of choosing gestures to suggest to an interaction designer seems desirable as well as practical.

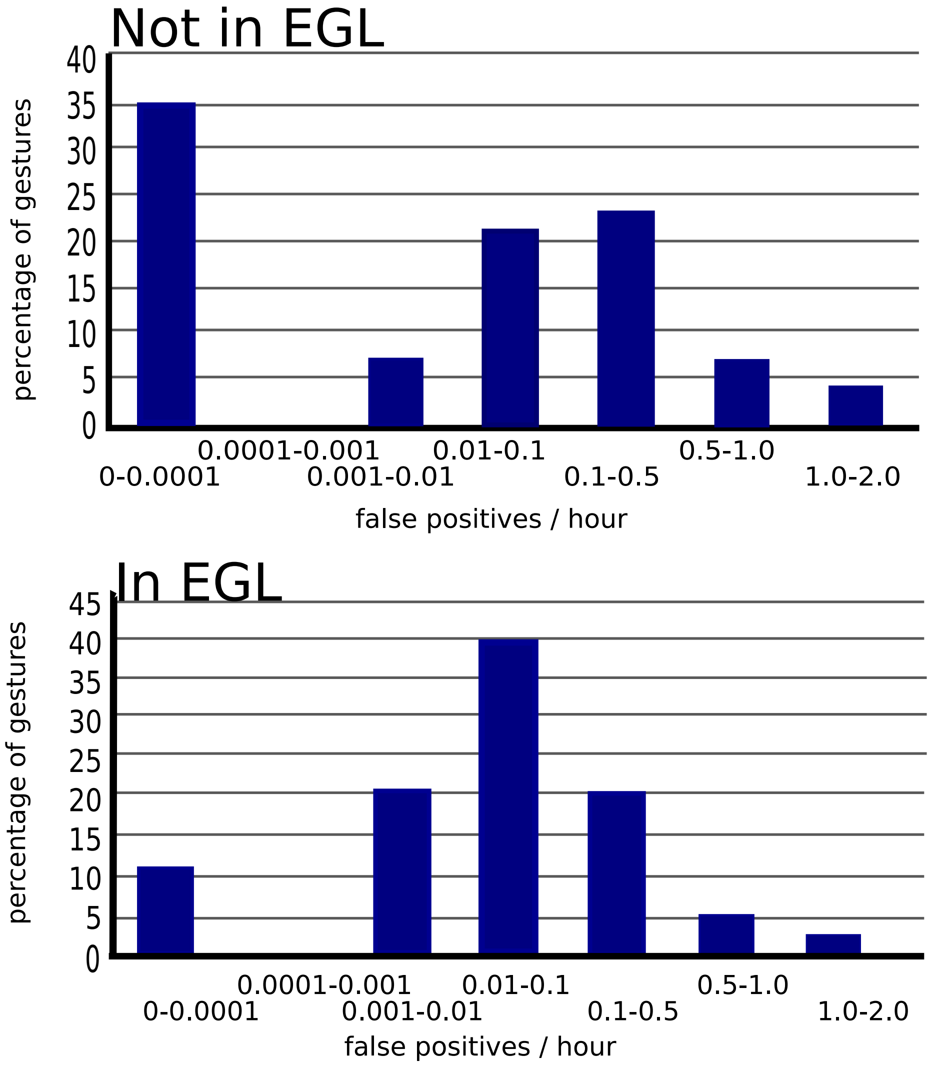


Figure 21: Histogram demonstrating the percentages of the number of false positives per hour for gestures with SAX representations not in the EGL (top) and all gestures with SAX representations in the EGL (bottom).

8. Future Work

To date, the task for most gesture recognition systems has been to optimize accuracy given a set of gestures to be recognized. In this paper, we have reversed the problem, seeking to discover which gestures might be most suitable for recognition.

However, improved suggestion ordering is an area for improvement. Performability might be improved by modeling how gestures are produced (Cao and Zhai, 2007) and prioritizing those gestures with least perceived effort. For domains where the coupling between sensor data and limb movement are not as apparent, such as accelerometer-based motion gestures, inverse kinematic models and 3D avatars seem appropriate both for prioritizing suggestions and for visualizing the gesture for the interaction designer. For situations with many degrees of freedom, such as whole body movement as tracked by the Microsoft Kinect©, the space of potential gestures may be extremely large. Physical and behavioral constraints might be applied to reduce the search space for the interaction designer. While MAGIC and MAGIC Summoning have been applied to multiple domains, we have only applied the gesture suggestion functions to trackpads. We are eager to investigate MAGIC Summoning's usefulness and usability in other domains.

9. Conclusion

We have described two pattern recognition tasks that can be used to help interaction designers create gesture interfaces: testing a user-defined gesture (and its classifier) against a previously captured database of typical usage sensor data to determine its tendency to trigger falsely and suggesting gestures automatically to the designer. We have shown that iSAX can be used to provide near immediate feedback to the user as to whether a gesture is inappropriate. While this method is approximate and recovers only a fraction of the total false positives in the EGL, MAGIC Summoning's results correlate strongly with those of HMMs, DTW, and the $1 Recognizer and can thus be used to provide guidance during training. We showed that MAGIC Summoning and the EGL could be used to create a null class of close false matches that increase the performance of the chosen classifier.

To suggest gestures to the interaction designer that may have low chance of triggering falsely, we exploited the SAX representation used to index the EGL. MAGIC Summoning generates all the strings not in the EGL, converts the SAX strings back into a gesture visualization, and suggests appropriate gestures to the designer. MAGIC Summoning also outputs classifiers for the gesture, trained on synthetic data generated from the SAX string. Using the task of finding command gestures for Mac trackpads, we showed that the gestures generated by MAGIC Summoning have generally low false positive rates when deployed and that the classifiers output by the system were adequate to the task of spotting the gesture.

Even if iSAX search of an EGL is not a perfect predictor for the false positives of a gesture in every day usage, we find that the approximations are sufficient to speed interface design significantly. MAGIC's methods are not intended to replace user testing with the final device. However, we believe that the tool will decrease the number of iterations needed to build a fast and stable gesture recognition interface.

Acknowledgments

This material is based upon work supported, in part, by the National Science Foundation under Grant No. 0812281. We would also like to thank Google for their support of the most recent advances in this project. Thanks also to David Quigley for sharing his Android EGL data set and Daniel Ashbrook for his original MAGICal efforts and collaborations.

References

Dan Ashbrook. *Enabling Mobile Microinteractions*. PhD thesis, Georgia Institute of Technology, Atlanta, Georgia, 2009.

Daniel Ashbrook and Thad Starner. MAGIC: a motion gesture design tool. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 2159–2168, New York, New York, 2010.

Mohammed Belatar and François Coldefy. Sketched menus and iconic gestures, techniques designed in the context of shareable interfaces. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 143–146, New York, New York, 2010.

Xiang Cao and Shumin Zhai. Modeling human performance of pen stroke gestures. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1495–1504, New York, New York, 2007.

Chi Tai Dang and Elisabeth André. Surface-poker: multimodality in tabletop games. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 251–252, New York, New York, 2010.

Roger Dannenberg and Dale Amon. A gesture based user interface prototyping system. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 127–132, New York, New York, 1989.

Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. a CAPpella: programming by demonstration of context-aware applications. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 33–40, New York, New York, 2004.

Jerry Fails and Dan Olsen. A design tool for camera-based interaction. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 449–456, New York, New York, 2003.

Ada Wai-Chee Fu, Eamonn Keogh, Leo Yung Lau, Chotirat Ann Ratanamahatana, and Raymond Chi-Wing Wong. Scaling and time warping in time series querying. *The International Journal on Very Large Data Bases*, 17(4):899–921, 2008.

François Guimbretière and Terry Winograd. Flowmenu: combining command, text, and data entry. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 213–216, 2000.

Richard Hamming. Error detecting and error correcting codes. *Bell Systems Technical Journal*, 29: 147–160, 1950.

Yuichi Hattori, Sozo Inoue, and Go Hirakawa. A large scale gathering system for activity data with mobile sensors. In *Proceedings of the IEEE International Symposium on Wearable Computers*, pages 97–100, Washington, District of Columbia, 2011.

Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. Direct manipulation interfaces. *Human-Computer Interaction*, 1(4):311–338, December 1985. ISSN 0737-0024.

Daniel Kohlsdorf, Thad Starner, and Daniel Ashbrook. MAGIC 2.0: A web tool for false positive prediction and prevention for gesture recognition systems. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pages 1–6, Washington, District of Columbia, 2011.

Danile Kohlsdorf. Motion gesture: False positive prediction and prevention. Master's thesis, University of Bremen, Bremen, Germany, 2011.

Yang Li. Gesture search: a tool for fast mobile data access. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 87–96, New York, New York, 2010.

Jessica Lin, Li Wei, and Eamonn Keogh. Experiencing sax: A novel symbolic representation of time series. *Journal of Data Mining and Knowledge Discovery*, 15(2):107–144, 2007.

Chris Long. *Quill: A Gesture Design Tool for Pen-based User Interfaces*. PhD thesis, University of California, Berkeley, California, 2001.

Kent Lyons, Helene Brashear, Tracy Westeyn, Jung Soo Kim, and Thad Starner. GART: the gesture and activity recognition toolkit. In *Proceedings of the International Conference on Human-Computer Interaction: Intelligent Multimodal Interaction Environments*, pages 718–727, Berlin, Germany, 2007.

Dan Maynes-Aminzade, Terry Winograd, and Takeo Igarashi. Eyepatch: prototyping camera-based interaction through examples. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 33–42, New York, New York, 2007.

Tom M. Mitchell. *Machine Learning*. McGraw Hill, New York, New York, 1997.

Tom Ouyang and Yang Li. Bootstrapping personal gesture shortcuts with the wisdom of the crowd and handwriting recognition. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 2895–2904, 2012.

Antti Pirhonen, Stephen Brewster, and Christopher Holguin. Gestural and audio metaphors as a means of control for mobile devices. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 291–298, New York, New York, 2002.

Jin Shieh and Eamonn Keogh. iSAX: indexing and mining terabyte sized time series. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 623–631, New York, New York, 2008.

Jin-Wien Shieh. *Time Series Retrievel: Indexing and Mapping Large Datasets*. PhD thesis, University California, Riverside, California, 2010.

Thad Starner, Joshua Weaver, and Alex Pentland. Real-time American Sign Language recognition using desk and wearable computer-based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371 – 1375, December 1998.

Tracy Westeyn, Helene Brashear, Amin Atrash, and Thad Starner. Georgia Tech Gesture Toolkit: supporting experiments in gesture recognition. In *Proceedings of the International Conference on Multimodal Interfaces*, pages 85–92, New York, New York, 2003.

Hendrik Witt. *Human-Computer Interfaces for Wearable Computers*. PhD thesis, University Bremen, Bremen, Germany, 2007.

Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 159–168, New York, New York, 2007.

Hee-Deok Yang, Stan Sclaroff, and Seong-Whan Lee. Sign language spotting with a threshold model based on conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1264–1277, 2009.