MAGIC 2.0: A Web Tool for False Positive Prediction and Prevention for Gesture Recognition Systems

Daniel Kohlsdorf
TZI
University Bremen
Bremen, Germany
Email: dkohl@tzi.de

Thad Starner
GVU Center
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332
Email: thad@gatech.edu

Daniel Ashbrook
Nokia Research Center Hollywood
Santa Monica, CA 90404
Email: daniel.ashbrook@nokia.com

Abstract

False positives are a common problem for interfaces that rely on gesture recognition. Often a gesture can seem fine in development but is found to trigger accidentally during an initial deployment of the interface, restarting development and increasing expense. In this work we introduce MAGIC 2.0, a technique for false positive prediction and prevention that can be used interactively during the interface design process. To ground our research, we implement MAGIC 2.0 as a web service and develop gesture interfaces using sensors on common Android mobile phone platforms. We use iSAX (indexable Symbolic Aggregate approXimation) to enable interactive searching (<2 sec/example) of a large database (>1,500,000 sec) of everyday user movements on a standard workstation to determine if a candidate gesture will trigger accidentally during use of an interface. We perform a user-independent study that suggests that the number of matches to this Everyday Gesture Library (EGL) database is indeed predictive of a candidate gesture's suitability. We compare iSAX to hidden Markov models (HMMs) and nearest neighbor with respect to accuracy and speed for the EGL search. Using iSAX on the EGL, we also develop a "garbage" class and show that including this class in recognition reduces errors.

1 Motivation and Related Work

Gestures are part of everyday life. They support human communication and are becoming a quick and easily accessible computer input method [16, 17, 19]. Ashbrook and Starner [2, 4] describe using gestures to create "micro interactions" where the user can perform functions on a mobile device. They argue that interfaces that are quick to access (under two seconds) and can be used for small, incremental interactions (approximately four seconds) allow users to interact more frequently with mobile devices while on-the-go [3, 14, 18]. Despite some significant successes by the Nintendo Wii and Apple iPod projects, few devices use motion-based gestures. One of the reasons is that building a gesture input system requires detailed pattern recognition knowledge which most interaction designers do not possess [9]. Another obstacle is the testing of the gesture system in everyday life. Everyday life movements like shaking hands or walking can be confused with designed gestures (for example, "shake to shuffle" on the iPod). Even interaction designers with a background in pattern recognition have difficulty in designing gestures that do not suffer from false positives in everyday life [2]. Running tests "in-the-wild" to determine the viability of a gesture is challenging and costly. If such testing shows the gestures to suffer from accidental triggering, then the gesture set has to be redesigned, and the tests run again. MAGIC 2.0 helps shorten this development cycle by predicting false positives.

In related work, the Gesture Interface Designer (GID) [5] allows users to design a physical interface on a screen that responds to pointing and finger gestures. Long's Quill [13], a pen gesture system, enables users to create pen gestures by example. Furthermore, Quill offers an aid for improving recognition. However, Long discovered that many Quill users do not understand basic error sources and have difficulty understanding suggestions from the aid system. SUEDE [12], another design tool, focuses on speech-based user interfaces. It supported a "design / test / analyze" iterative work flow that inspired MAGIC. Other existing tools include Crayons [9], Eyepatch [15] and aCapella [7], but none focus on false positives and few allow direct deployment (see Ashbrook [2] for an overview of state-of-the-art gesture tools). Numerous machine learning tools like Weka [8] and GART (which is in turn built on Cambridge's Hidden Markov Model Toolkit) [11, 6] are often used for gesture recognition but act more as a library than a design tool for interaction designers.

Ashbrook and Starner [2, 4] introduced a gesture creation and testing tool called Multiple Action Gesture Interface Creation tool short (MAGIC) for motion gestures from wrist mounted accelerometers. Unlike the above tools, MAGIC combines interactive recognizer building with a false positive testing method. MAGIC encourages **iterative** design and attempts to predict the number of false positives by a module called "Everyday Gesture Library" (EGL). The EGL is a large database of users' movements recorded from everyday life. After the designer has created a gesture MAGIC searches the EGL for similar patterns as potential false positive sources using the Nearest Neighbor method with a Dynamic Time Warping (DTW) distance measure [1]. The potential false positive "hits" are returned to the designer, indexed with the times in the EGL where the hit occurred.

One of the disadvantages of the previous version of MAGIC is the time required for computing the hits in the EGL, resulting in delays for searches on large EGLs. Such a delay forces a batch approach for searching the EGL, resulting in users not checking against the EGL when creating each gesture. For MAGIC 2.0 we desire a system fast enough that it can search the EGL as each gesture is created. We evaluate the possibility of utilizing multi-dimensional iSAX (indexable String ApproXimation) as a fast method for EGL search. Furthermore we describe a method of creating a "garbage" class from the hits in EGL and demonstrate its effectiveness at helping reduce false positives in a user-independent test of a gesture set.

The original MAGIC assumed that high false positive rates of a gesture during EGL search indicate high false positive rates in everyday life. If this hypothesis is true, a designer can delete gestures that trigger many false positives early in gesture design. We test this assumption empirically. (Note that utilizing the EGL will not obviate real life testing but increases the chance of a success so that less expensive real life tests are necessary.) We present a first prototype of MAGIC 2.0 grounded by the task of creating a gesture set for Android phones using their built-in accelerometer. While our experiments focus on accelerometer based motion gestures MAGIC 2.0 can handle all kinds of gestures recorded as a time series of any dimension.

2 False Positive Prediction and Prevention

In this section we review iSAX [10] for the convenience of the reader. Later we will show how to extend iSAX for multiple dimensions and how to use iSAX for false positive prevention and prediction.

2.1 iSAX indexing

iSAX indexes time series. Given a query, iSAX returns candidates which are close to that query. In order to build an index structure of a set of time series, each time series is first converted into a Piecewise Aggregate Approximation (PAA). The PAA compresses a time series $T = t_1, t_2, ...t_n$ of length n to a user specified length N by applying the following formula:

$$\bar{t_i} = \frac{N}{n} \sum_{j=n/N(i-1)+1}^{n/N*i} t_j \tag{1}$$

PAA divides the time series in N blocks and saves the mean of the block resulting in a time series $\bar{T}$ of length N.

The second step is to compute an index key from the PAA representation using SAX(String ApproXimation) with a user specified parameter alphabet size a. To do string conversion, imagine the y-axis divided into different regions, bounded by breakpoints. Each of these regions is associated with a symbol from an alphabet A of size a. If a value from our PAA representation exceeds a break point, it is replaced by the symbol referring to the region above this break point. The result is a string of length N from the alphabet A. Consider the time series $T = [1, 4, 1, 4, 1, 2, 1, 2, 6, 8]$, the breakpoints $1, 2, 7$ and the alphabet $A = A, B, C$. The PAA of word length 5 is then $PAA(5, ts) = [2.5, 2.5, 1.5, 1.5, 7]$. The SAX word would be: "BBAAC". Note that the symbols do not have to be characters; later we will use numbers. If a time series is Gaussian distributed we can obtain equiprobable characters. We choose the regions on the y-axis in a way that the area under a normal distribution N(0,1) is $\frac{1}{a}$. Such break points can be looked up in a statistical table. The overall SAX process is shown in Figure 1.

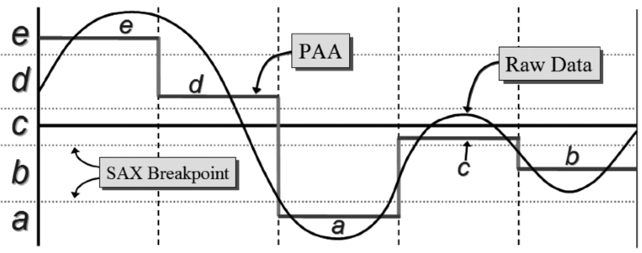

Figure 1. SAX used to convert a time series to a string.

It is also possible to create words with different cardinalities per symbol. Such a word will be written as symbol cardinality tuples. The string: 6_8.5_7.5_7 means the SAX word: $6, 5, 5$ with the cardinalities $8, 7, 7$. In the first word position 8 symbols are possible, but in the following two words only 7 are possible. (In the original paper the authors use binary strings of different length to represent the value, cardinality pair) The last step is to build the index structure by inserting all time series in a “iSAX Tree”.

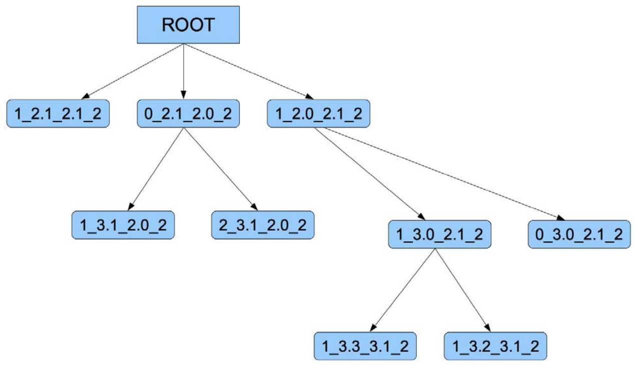

Figure 2. iSAX tree construction.

In the beginning of construction the only node is the root node. A node saves a hash table mapping of all successors to a SAX word and an array of alphabet sizes with $length(array) = word\_length$. In the root node it is an array of the given base cardinality.

Each time series inserted is converted to a SAX representation first. Then the SAX string is looked up in the current nodes hash table. If it does not exist a new node is created, and the time series is inserted in this node. If it exists the time series is simply inserted in this node. All the above operations are performed on a node type called the terminal node or leaf. Each leaf points to a bucket or file with the referring time series. If the number of time series in a terminal node exceeds the threshold th, the overfull terminal node is deleted and, the cardinality is increased in one array position (by a round robin process). Furthermore, a new internal node is created with the hash key of the old node and the new cardinality. All time series of the old node are swapped to the new node. In this way, the time series of the old node will split into multiple buckets because, in a bigger alphabet, the SAX word will fit more exactly to the time series.

2.2 False Positive Prediction and Prevention Using iSAX

We speed the EGL search by inserting all time series (collected from a large data set of different users wearing the required device) into an n-dimensional iSAX tree, where n is the length of the feature vector. Such a tree consists of n sub trees each storing the time series of one dimension (see Figure 3).

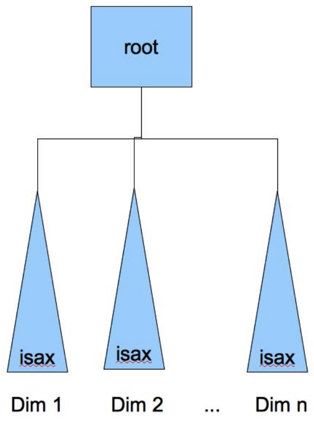

Figure 3. A multi dimensional iSAX tree.

Only “interesting” regions are considered, where interesting means the sample variance is over a user specified threshold [2]. For a new gesture, the tree is searched in all n trees. The result of this search are n files of time series, one for each dimension. A hit within the EGL is when regions matching the query overlap in each dimension.

Because comparing time stamps can be expensive to compute, an approximation is introduced. Our approximation is that there can never be more hits than time series in the minimum file. For example, consider the result of the 3-dimensional iSAX search is: $x = 4$, $y = 20$ and $z = 6$. There can never be more then 4 hits when comparing overlapping regions.

iSAX keeps similar time series in a file at a root node. The number of overlapping time series from these files (hopefully) predicts the number of false positives as compared to a more exact search with the classifier.

Another use of the EGL is to train garbage classes. In continuous activity recognition, most classifiers require a threshold. For Nearest Neighbor, the threshold is set as the maximum allowable distance from an example to still be considered to be similar to that example (and the example’s class). For Hidden Markov Models, the threshold is the minimum likelihood required to match a class. However, finding a threshold that does not cause false negatives but prevents most false positives is difficult. One possible approach is to find a training set for potential false positives and ignore classifications that fall into this class. With MAGIC 2.0, one source for such a training set are the hits from the EGL. In the following section we will describe our experiments with Nearest Neighbor, iSAX and Hidden Markov Models to show that accidental triggering in everyday life can be predicted and prevented.

2.3 Experimental Verification

Does the number of hits from an approximate search using iSAX provide an indication of the amount of false triggering that might happen in everyday life using common classifiers? To investigate this question, we perform two experiments where we design four gestures (circle, shake, touching shoulder, and hacking) intended to be used with Android phones. The first compares the number of hits returned by iSAX from the EGL to the number returned by 1-Nearest Neighbour(1-NN) and Hidden Markov Models(HMM). The second compares these values to the number of false positives that occur when four novel users attempt to use the gesture system during their everyday lives.

We collected a large EGL ($>$ 1.5 million seconds or 19 days total) from six users' phones running the Android OS. One author acted as an interaction designer and created four gesture classes with 12 examples each using MAGIC 2.0. By selecting appropriate thresholds, the designer achieved 93% accuracy on the training set with 1-NN and 100% accuracy with HMMs (using an eight state model with one skip transition). Next we searched the EGL for each of the examples in each of the classes using iSAX, 1-NN (using DTW), and HMMs. The iSAX parameters were estimated empirically:

1. **word length**: 4
2. **base card**: 1
3. **bucket**: 6000

As shown in Figure 2.3, iSAX returns much fewer hits than 1-NN and HMMs. However, the magnitude of the iSAX values correlate strongly with the 1-NN ($r^2 = 0.92$) and HMM ($r^2 = 0.94$) results. Thus, when the iSAX EGL search returns a high number of hits, the designer should replace the gesture. On the other hand, gestures that return a low number of EGL hits with iSAX are good candidates for user testing. The times needed to run an EGL search on a standard laptop were 8 hours for HMMs, 10 - 15 minutes for 1-NN, and 22 seconds with iSAX. Given the highly parallel nature of the task, iSAX search times can be reduced to a few seconds on a modern server.

Next, we performed a four subject study to determine how well MAGIC predicts and prevents false positives with users who did not contribute to the EGL nor to the training sets for the gestures. The users were taught the gestures by the experimenter. The users were asked to perform each gesture 10 times, and we recorded the recognizer's accuracy of recognizing this training set. Afterwards, users trained with the gesture system, using feedback from the recognizer to adapt how they performed their gestures to achieve better accuracy. This approach simulates current commercial gesture systems (like on the Wii), which use avatars

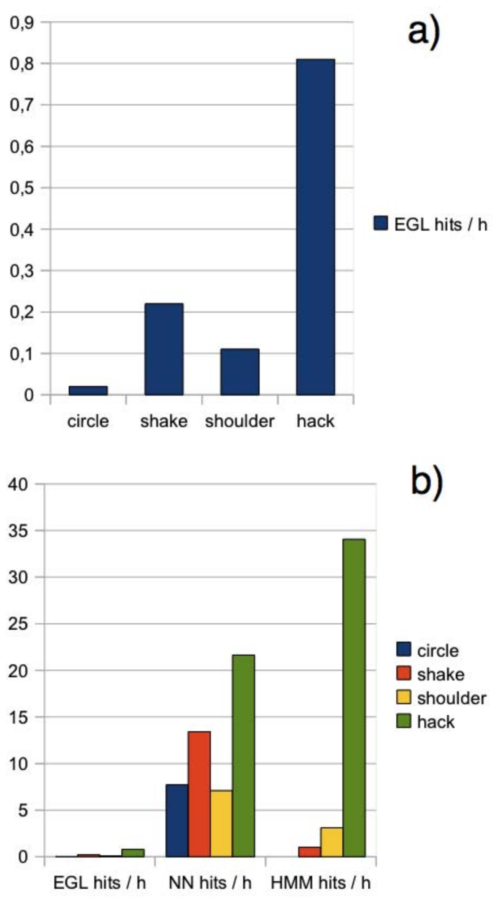

Figure 4. a) The hits per hour in the EGL based on iSAX search. b) A comparison on the number of hits per hour returned by iSAX, 1-NN and HMMs from the EGL.

to help users learn the gestures. The users' performance on the gestures was determined again. Users were able to improve from approximately 60% to 95% accuracy. After this training session, we installed software on the users' phones that notifies the users to perform one of the gestures every hour, selected randomly. All activities (gestures and everyday life) were recorded. We used both 1-NN and HMMs with thresholds as set during the designer's training (maximum likelihood optimized inter-class variability) to spot and recognize the gestures in the four subjects' data streams, recording false positives as well. 1-NN found all the intentional gestures, while HMMs found $50\% - 70\%$. The iSAX predictions (from the EGL during the design process) on the number of false positives had a high correlation to what was observed in practice, with r^2 of 0.7 for 1-NN and 0.94 for HMMs (see Figure 2.3).

We performed a second variation of this test, exploring the effectiveness of using a garbage class trained from the EGL with the HMMs. For the 1-NN classifier, we adjusted thresholds upward to avoid misclassifications in the designer's EGL while still detecting the gestures from the training set. Figure 2.3 shows the number of additional false positives per hour this technique avoids when applied to the four users' "in-the-wild" data versus thresholding based solely on inter-class distance between the original four gestures. Using the EGL to tune the recognizers resulted in a statistically significant improvement for both 1-NN (approximately 11 additionally rejected false positives per hour, $p << 0.0001$) and HMMs (also 11 additionally rejected false positives per hour, $p < 0.05$). Gesture recognition accuracy and correlation to the iSAX EGL hits remained consistent with the first variation of the experiment.

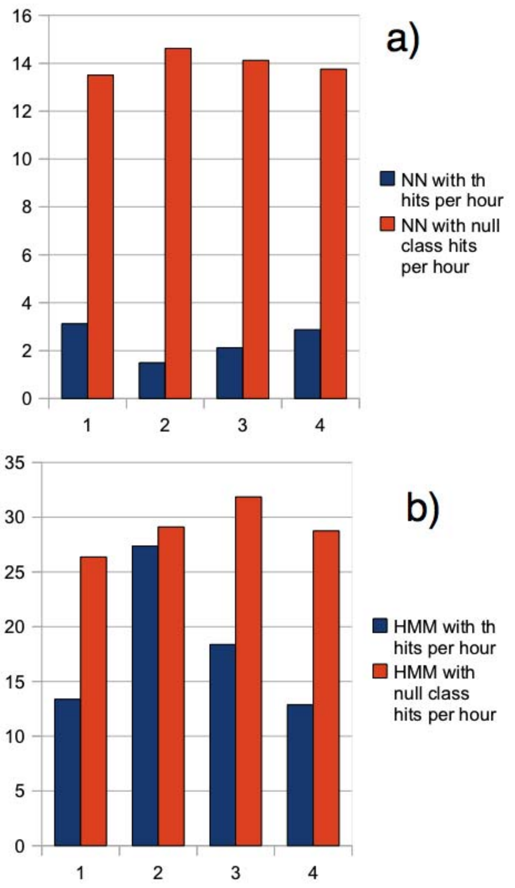

Figure 5. a) The false positives per hour avoided using an inter-class threshold versus tuning the threshold upward based on EGL hits b) The false positives per hour avoided for HMMs using a inter-class threshold versus the inclusion of a garbage class with no minimum likelihood threshold.

3 MAGIC 2.0

The MAGIC2.0 web application prototype includes basic functionality like managing users, creating projects for new gesture sets and creating gestures. In this case, a prebuilt EGL iSAX tree is included (see Section 2.2) for accelerometers on Android phones. Gestures can be evaluated for intra-class variability, distinctiveness between classes, and closeness to movements recorded in the EGL. Once a designer settles on a suitable gesture set, a HMM or 1-NN recognizer can be output.

The designer uploads a set of examples for each gesture to the web server. Once uploaded, the web server classifies each example based on its current models and searches for hits in the EGL. The classification and the EGL hits are stored in a database. The examples are listed in a sidebar under the gesture to which they refer. A red cross next to the example in the sidebar indicates that it was misclassified. A green circle indicates correct classification. By clicking on an example, the time series is plotted and the hits with the Everyday Gesture Library are shown (see Figure 6).

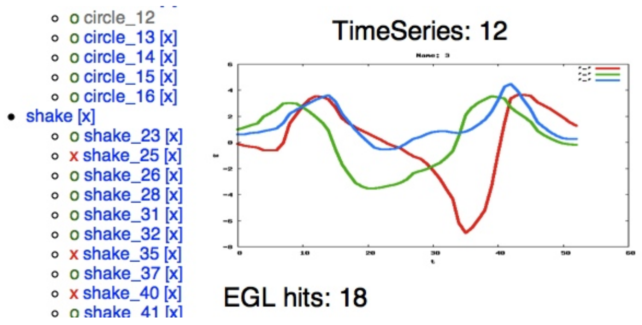

Figure 6. Training examples for a gesture.

By clicking on a gesture in the sidebar, two mean and standard deviation distance plots are displayed. In one plot the means and standard deviations of the distances from all examples in a class to all other examples of that class are shown (see Figure 7). In the other, the means and standard deviations of the distances between classes are displayed. These visualizations help expert designers gain insight as to the consistency and distinctiveness of a set of gestures.

4 Conclusion

Our results suggest that iSAX searching of the Everyday Gesture Library database is indeed sufficient to screen a candidate gesture to warn of large numbers of false positives when deployed "in-the-wild" with 1-NN and HMM recognizers. False positive rates can be significantly reduced by training classifiers with the results of a garbage class determined by the iSAX results. iSAX proved to be much

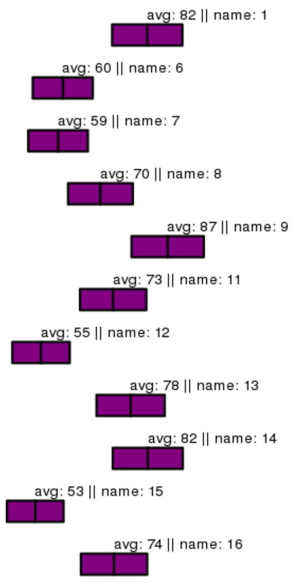

Figure 7. Mean and standard deviation of the distance between each example in a class and the class as a whole.

faster then 1-NN or HMM classification and seems sufficiently fast to enable interactive gesture design in MAGIC 2.0. However, it is not clear how sensitive the system is to iSAX parameters; a sensitivity analysis will be required to determine if the system can be generalized or can be autotuned for each new project.

Furthermore, we wish to test the MAGIC concept with other pattern recognition methods (for example, Support Vector Machines, Decision Trees, etc.). We are also integrating the slower search methods in the web application as a background service (so that designers, once they determine that a given gesture is worth pursuing, can use the actual recognition algorithm to better estimate the false positive rate).

We presented the MAGIC 2.0 web-based tool for gesture design and evaluation and applied it to smart phone gestures. We discussed a fast way to predict the false positive rate of gestures and even prevent them. We verified our approach experimentally with a user independent study and are preparing to release the tool publicly.

5 Acknowledgements

Thanks to ETRI, Google, and NSF for their support of this project, and David Minnen for the use of his SAX creation picture.

References

[1] e. a. Ada Wai-Chee Fu, Eamonn Keogh. Scaling and time warping in time series querying. *The International Journal on Very Large Data Bases*, 2008.

[2] D. Ashbrook. *Enabling Mobile Microinteractions*. PhD thesis, georgia institute of technology, 2009.

[3] D. Ashbrook, J. Clawson, K. Lyons, T. Starner, and N. J. Patel. Quickdraw: the impact of mobility and on-body placement on device access time. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 219–222, 2008.

[4] D. Ashbrook and T. Starner. Magic: a motion gesture design tool. In *CHI*, pages 2159–2168, New York, NY, USA, 2010. ACM.

[5] A. Dannenberg. A gesture based user interface prototyping system. *Proceedings UIST 89*, 1989.

[6] e. a. Everman, Gales. *The HTK Book*. Cambridge University Engeneering Department, 2005.

[7] e. a. Hamid, Beckmann. a capella: programming by demonstration of context aware applications. *Proceedings CHI 2004*, 2004.

[8] E. F. Ian H. Witten. *Data Mining. Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

[9] D. R. O. Jerry Allan Fails. A design tool for camera-based interaction. *Proceedings CHI 2003*, 2003.

[10] E. K. Jin Shieh. isax: Indexing and mining terabyte sized time series. *SIGKDD 2008*, 2008.

[11] e. a. Kent Lyons, Helene Brashear. Gart: The gesture and activity recognition toolkit. *Proceedings of the 12th International Conference on Human-Computer Interaction*, 2007.

[12] e. a. Klemmer, Sinha. Suede: a wizard of oz prototyping tool for speech user interfaces. *Proceedings UIST 00*, 2000.

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

[14] K. Lyons, C. Skeels, T. Starner, C. M. Snoeck, B. A. Wong, and D. Ashbrook. Augmenting conversations using dual-purpose speech. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 237–246, New York, NY, USA, 2004. ACM Press.

[15] I. Maynes-Aminzade, Winograd. Eyepatch: prototyping camera-based interaction through examples. *Proceedings UIST 07*, 2007.

[16] A. Pirhonen. Gestural and audio metaphors as a means of control for mobile devices. *Proceedings CHI 2002*, 2002.

[17] T. Starner. Visual recognition of american sign language using hidden markov models. Master's thesis, Massachusetts Institute of Technology, 1995.

[18] T. E. Starner, C. M. Snoeck, B. A. Wong, and R. M. McGuire. Use of mobile appointment scheduling devices. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1501–1504, New York, NY, USA, 2004. ACM Press.

[19] H. Witt. *Human-Computer Interfaces for Wearable Computers*. PhD thesis, University Bremen, 2007.