Hand Grip Pattern Recognition for Mobile User Interfaces

Kee-Eung Kim, Wook Chang, Sung-Jung Cho, Junghyun Shim,
Hyunjeong Lee, Joonah Park, Youngbeom Lee, and Sangryong Kim
Interaction Lab
Samsung Advanced Institute of Technology
P.O.Box 111
Suwon, Korea 440-600
{kekim, wook.chang, sung-jung.cho, id.shim, hyem.lee, joonah, leey, srkim}@samsung.com

Abstract

This paper presents a novel user interface for handheld mobile devices by recognizing hand grip patterns. Particularly, we consider the scenario where the device is provided with an array of capacitive touch sensors underneath the exterior cover. In order to provide the users with intuitive and natural manipulation experience, we use pattern recognition techniques for identifying the users' hand grips from the touch sensors. Preliminary user studies suggest that filtering out unintended user hand grip is one of the most important issues to be resolved. We discuss the details of the prototype implementation, as well as engineering challenges for practical deployment.

Figure 1: Tools, mobile devices, and grip patterns

Introduction

Recent advances in mobile user interface research typically involve various types of sensors in order to facilitate natural interaction between the user and the mobile device.

Hinckley *et al.* (2000) used an infrared sensor, a 2-axis accelerometer and multiple capacitive touch sensors on a PDA to detect user gestures such as initiating voice memo, changing the display orientation (portrait or landscape), scrolling through the displayed content, and switching the power on or off. Poupyrev *et al.* (2002) explored the idea of using a 2-axis accelerometer and piezoceramic actuator in scrolling through the text and selecting a displayed item.

Although not in the domain of mobile devices, Rekimoto (2002) and Shen *et al.* (2004) presented instrumented tables with a mesh-shaped array of capacitive touch sensors in order to detect the users' hand positions and shape. These tables allow users to manipulate displayed objects on the table with various intuitive hand gestures.

This paper presents a novel mobile device interface using an array of capacitive touch sensors to identify the hand grip pattern and launch appropriate applications. Specifically, we build upon the idea that there are natural grip patterns — affordances (Norman 1988) — when using handheld tools. Most of the tools invented so far are designed in appropriate shapes to induce particular grip patterns (Figure 1). Hence, humans have learned to interact with tools in the most convenient way over the life-long experiences. We observed the same behavior when using handheld mobile devices. Some mobile devices are designed in their exterior shapes and button layouts to induce specific grip patterns when using specialized mobile applications such as taking pictures, gaming, *etc*. Other mobile devices are designed in more generic shapes, but users typically adapt themselves while learning to use specific applications. Examplar behaviors include using two hands when composing a text message. We conjecture that the later type of mobile devices and user adaptations will be more ubiquitous as the mobile devices available in the market get more complex and multi-functional. Hence, it is crucial to provide intuitive and easy-to-use interfaces, and we decided to focus on recognizing the grip patterns to understand the user intentions.

Specifically, We are interested in finding the most natural grip patterns for the pre-defined set of mobile applications, and execute the applications when the touch sensor signals match the pre-defined set of hand grip patterns. We use pattern recognition algorithms (Duda, Hart, & Stork 2001) for classifying the touch sensor signals.

One of the important discoveries from our approach is that we have identified some critical engineering issues to be resolved in order to make the system practical. Besides obtaining the well-defined and easy-to-classify hand grip pattern set, designing the reliable classifier that can pass on unintentional skin touch is very important as well for the successful commercialization. We present the details of our prototype implementation including the hardware design, the software development, and the classification experiments.

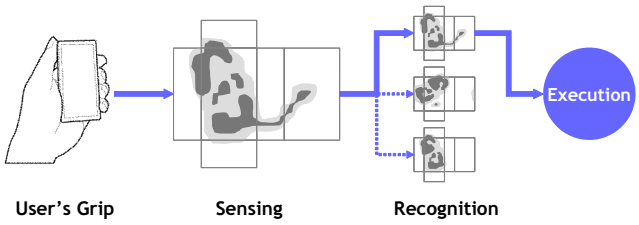

Figure 2: Basic interaction flow

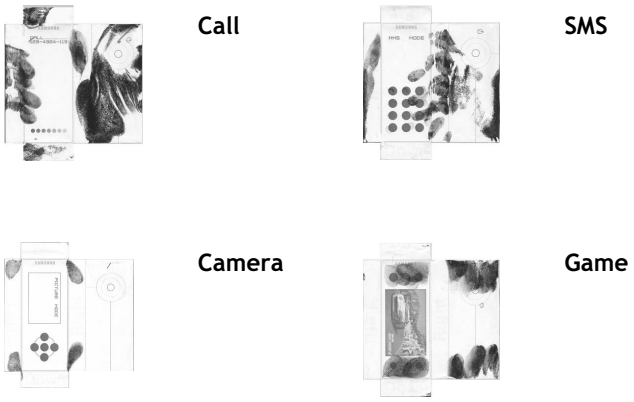

Figure 3: Hand grip images from painted gloves

Overview of the System

Figure 2 depicts the conceptual interaction flow when using the proposed system. As the user grips the mobile device, touch region information is gathered from the touch sensors. This raw data is then preprocessed to enhance and stabilize the data, and classified into one of the pre-defined grip pattern classes for launching appropriate mobile device applications.

In order to verify our hypothesis that the hand grip patterns are discernible for a carefully defined set of mobile device applications, we prepared a mock-up by carving out a piece of hard styrofoam and collected the hand grip data by letting 9 subjects hold the mock-up with painted gloves for 4 different hypothetical applications (Figure 3). A total of 36 grip images were scanned and then a decision tree was built upon the image data. We obtained the recognition accuracy of 56% even though the image was very noisy.

After confirming the feasibility with the preliminary result, we prepared a number of different working prototype systems, experimenting with different kinds of touch sensors. The prototype system we report in this paper is based on capacitative touch sensors from ESSD [1]. The ESSD SS01 8-channel sensor chip produces independent touch signals that ranges from 0 to 255. Note that the signals are not absolute values — they are relative in the sense that the values vary depending on the operating conditions such as the humidity of the environment, the design of the sensor electrodes, and the material between the skin and the electrodes. We used 8 sensor chips to handle 64 touch sensor electrodes.

[1] www.essd.com

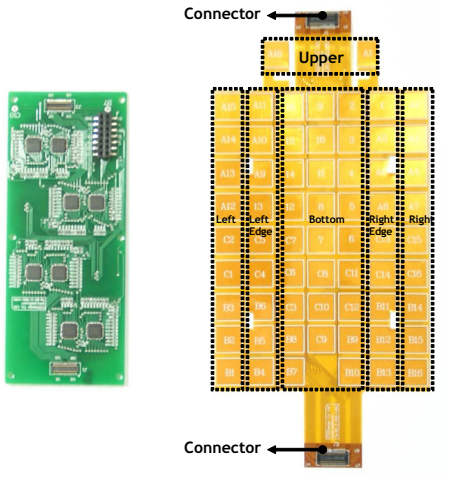

Figure 4: The sensor board with 8 ESSD SS01 touch sensor chips and the flexible PCB with 64 touch electrodes

Figure 5: The exterior photos of the prototype device

The sensor electrodes were etched on a flexible PCB and glued beneath the plastic cover of the prototype system (Figure 4). By trial and error, we figured out that electrodes with sizes 8mm $\times$ 8mm were adequate for sensing the contact of the hand with enough resolution and, at the same time, making reliable measurements through the 1.8 mm-thick plastic body. Figure 5 shows the photos of the prototype system.

Since we have to filter out unintentional user grips, such as picking up the device from the table, we also embedded a 3-axis accelerometer KXM52 from Kionix [2] to detect stability of the device. The idea here is that we let the user hold the device still for a short period of time in order to activate the grip pattern classifier. We later made this duration 0.4 second through usability evaluations.

We also included Samsung S3C2410X01 ARM CPU in the prototype system as the main processor. All the preprocessing, recognition, and user interface mock-up software modules were executed on the ARM CPU. The 2MB memory limit for storing and the 16MB memory limit for executing the code prevents us from deploying complex pattern recognition algorithms. We will get back to this issue in Section . The overall hardware schematic diagram of the system is shown in Figure 6.

[2] www.kionix.com

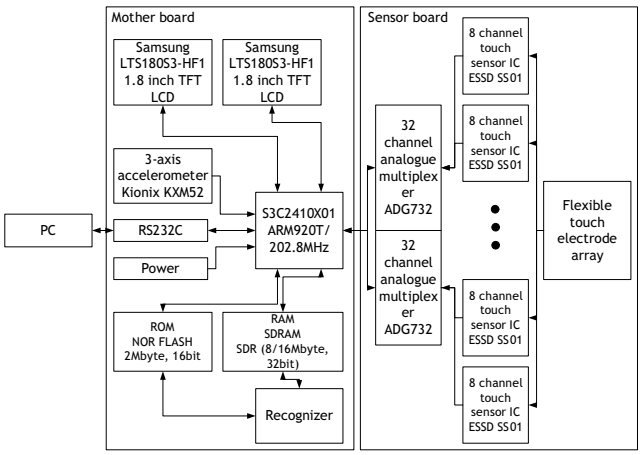

Figure 6: The hardware schematic diagram of the prototype device

Figure 7: Defining 8 grip patterns for 5 mobile device applications

Building the Hand Grip Classifier

Besides overcoming the hardware limitations and constraints by choosing the most appropriate touch sensor and designing the ideal sensor electrodes, building the hand grip classifier is important as well. In this section, we describe how we defined grip pattern classes for recognition, collect the grip pattern data for training, and implement the classifier.

Through careful studies including brainstorming sessions and user interviews, we selected the set of mobile device applications and the set of associated hand grip pattern classes. The 5 selected mobile device applications were

- `CALL`: receiving an incoming call,
- `SMS`: composing a text message,
- `CAMERA`: taking a picture,
- `VIDEO`: playing a video, and
- `GAME`: playing a game.

We assigned 2 grip patterns for `SMS` (one-handed and two-handed), and 3 for `CAMERA` (horizontally one-handed, vertically one-handed, and vertically two-handed), resulting in a total of 8 grip patterns (Figure 7). The mobile device applications were selected through the field study on frequency of usage for a typical multimedia capable mobile phone, and the associated grip pattern classes were defined by interviewing subjects to suggest the most natural hand grip when using the application.

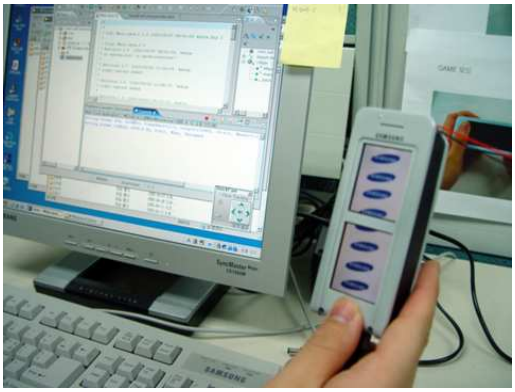

Figure 8: Collecting grip training data from subjects

After the grip pattern classes were defined, we collected the hand grip training data from 50 subjects. Each subject was asked to hold the prototype system 5 times for each grip pattern (Figure 8). The touch sensor chip continuously generated the data at 30Hz, and we averaged the 15 touch data (over 0.5 second) collected during the device was held still. Hence the training data from each grip trial was composed of 64 numeric fields for touch sensor readings. The final training data was composed of 250 instances for each of the 8 grip patterns.

We used the Weka (Witten & Frank 2005) machine learning library for the grip pattern classifiers. Specifically, we trained naive Bayes and support vector machine classifiers on the training data gathered through the process described above. All the training was done off-line — the classifier was trained on a desktop computer and then the parameters were extracted to implement the embedded versions of the classifiers (classification only) to be deployed into the prototype device.

Experiments

In this section, we report the classification accuracy of naive Bayes (NB) and support vector machine (SVM) classifiers on the training data. In the case of SVM, the Weka library uses sequential minimal optimization algorithm by Platt (1998). The multiple-ness of the classes is handled by combining pairwise two-class SVMs with the one-versus-one method, rather than the one-versus-all method. We trained the SVM with a variety of kernels, preprocessing filters, and complexity parameters. Figure 9 summarizes the results.

As we can see in the cross validation results, the overfitting is prevalent throughout the experiments. Apparently, we need more training data to obtain a reliable classification results. However, the cost (both time and effort) of acquiring the hand grip training data from subjects was prohibitive,

Classifier		Accuracy	CV
NB		79 %	76 %
SVM	L, N, C=1.0	93.6 %	91 %
	L, N, C=10.0	95.5 %	90.7 %
	L, S, C=1.0	96.1 %	90.5 %
	L, S, C=10.0	96.2 %	90.3 %
SVM	Q, N, C=1.0	97.3 %	90.8 %
	Q, N, C=10.0	99.6 %	90.2 %
	Q, S, C=1.0	100 %	91.8 %
	Q, S, C=10.0	100 %	91.8 %
SVM	R, N, γ=1.0, C=1.0	97.1 %	90.8 %
	R, N, γ=1.0, C=10.0	99.9 %	91.8 %
	R, S, γ=0.05, C=1.0	98.5 %	90.2 %
	R, S, γ=0.05, C=10.0	100 %	91.6 %

Figure 9: Classification accuracy results from naive Bayes (NB) and support vector machine (SVM). For SVMs, the kernels are linear polynomial (L), quadratic polynomial (Q), and radial basis function (R). The preprocessors used for attributes are normalization (N) by scaling the maximum value 1 and the minimum value 0, and standardization (S) by making the mean 0 and the standard deviation 1. We tried two different values 1.0 and 10.0 for the complexity parameter (C) in training SVMs. γ represents the width in the radial basis function kernels. We also show the results from the 10-fold cross validation (CV) with re-sampling.

so we decided to go ahead with the original data without employing more subjects to collect additional training data, since we were in the early stage of the prototype development.

Even though the accuracies are not at 100%, the classifiers mostly made correct answers on the training data. Figure 10 and Figure 11 show the typical confusion matrices of NB and SVM classifiers. Note that, in the case of SVM, if we take into account the fact that we only need differentiate among `CALL`, `SMS`, `CAMERA`, `VIDEO`, and `GAME`, the classifier yields only 1 error among 2000 training instances. Also, as expected, the most confusing pair of classes for the classifiers was the two one-handed grip patterns for `CAMERA`. This is because there is hardly a difference in touch areas except for the orientation of the prototype device.

We were also interested in the subject independence-ness of the grip pattern classifiers. Figure 12 shows the classification results from the subject-independence cross validation. We can also observe the classifiers being overfit. The leave-one-subject-out cross validation reduces the gap between the accuracy and the cross validation typically by 2$\sim$3 %, hence we expect that the cross validation results will improve as we employ more subjects for additional training data.

Porting the trained classifier to the prototype device was another issue. Since we have limited amount of memory for storing the parameters of classifiers, we could only consider simple classifiers. The NB classifier was no problem at all. However, as for the SVM classifiers, the prototype device could only store the classifiers with linear kernels — we do not have enough memory to store the large number of support vectors, but in the case of linear kernels, we only need to store the sum of the weights for support vectors. We refer to the implementation of the SMO Java class in (Witten & Frank 2005) for the details on compactly storing SVM classifiers with linear kernels.

Classifier		Accuracy	SI-CV
NB		79 %	76.6 %
SVM	L, N, C=1.0	93.6 %	88.4 %
	L, N, C=10.0	95.5 %	85.3 %
	L, S, C=1.0	96.1 %	85.1 %
	L, S, C=10.0	96.2 %	84.0 %
SVM	Q, N, C=1.0	97.3 %	85.3 %
	Q, N, C=10.0	99.6 %	84.9 %
	Q, S, C=1.0	100 %	85.4 %
	Q, S, C=10.0	100 %	85.4 %
SVM	R, N, γ=1.0, C=1.0	97.1 %	87.1 %
	R, N, γ=1.0, C=10.0	99.9 %	86.9 %
	R, S, γ=0.05, C=1.0	98.5 %	84.1 %
	R, S, γ=0.05, C=10.0	100 %	83.8 %

Figure 12: Same set of experiments as in Figure 9 except 10-fold subject-independence cross validation (SI-CV) results are shown.

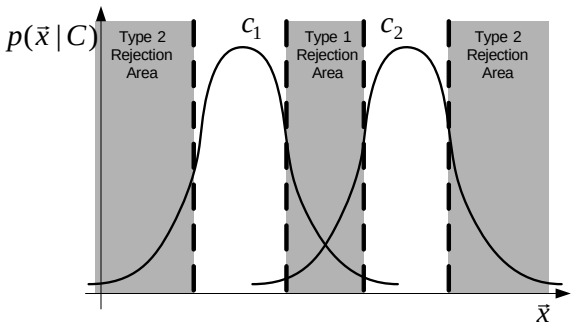

Figure 13: Two types of rejections to be handled by rejection classifier

Rejection Classifier

After the preliminary experiments with NB and SVM classifiers, we decided to focus on identifying unintentional hand grips. Without any rejection, the classifiers incorrectly try to classify every touch behavior such as fiddling inside the hand to grasp the device for a more comfortable position, accidental skin contact other than hands, or even the device not being contacted at all — the touch sensor chip yield relative numeric values, rather than explicit touch or non-touch status.

There are two types of rejections to consider, as illustrated in Figure 13. The first type is when the input data is approximately equal distance from more than one class. In this case, the differences in posterior probabilities of the candidate classes are small. This can be handled by traditional Bayesian decision theory (Chow 1970) by asserting that the differences in posterior probabilities are above a pre-defined

classified as →	a	b	c	d	e	f	g	h
a = CALL	229	3	4	0	12	1	1	0
b = SMS (one-handed)	6	201	20	2	20	0	1	0
c = SMS (two-handed)	5	9	209	2	24	1	0	0
d = CAMERA (horizontal, one-handed)	0	2	9	162	70	1	0	6
e = CAMERA (vertical, one-handed)	7	2	6	76	152	3	0	4
f = CAMERA (two-handed)	0	1	1	1	28	204	4	11
g = VIDEO	1	0	0	0	8	22	203	16
h = GAME	0	0	0	2	5	20	3	220

Figure 10: Confusion matrix for the naive Bayes classifier (accuracy = 79%)

classified as →	a	b	c	d	e	f	g	h
a = CALL	250	0	0	0	0	0	0	0
b = SMS (one-handed)	0	249	1	0	0	0	0	0
c = SMS (two-handed)	0	0	250	0	0	0	0	0
d = CAMERA (horizontal, one-handed)	0	0	0	213	37	0	0	0
e = CAMERA (vertical, one-handed)	0	1	0	39	210	0	0	0
f = CAMERA (two-handed)	0	0	0	0	0	250	0	0
g = VIDEO	0	0	0	0	0	0	250	0
h = GAME	0	0	0	0	0	0	0	250

Figure 11: Confusion matrix for the SVM classifier with linear kernel, standardization preprocessing and complexity parameter value 1.0 (accuracy = 96.1%)

threshold. The highlighted area between two peaks in Figure 13 is the rejection region of the first type.

The second type is a little more subtle, but important in our case, and the focus of the rest of this section. This is when the input data is of very large distance from any of the classes. This happens more frequently in our system since the training data covers very small subset of the whole input space — since the classifier is only trained on meaningful user hand grips, *i.e.*, we assume the data is generated only when the user intends to initiate interaction, the classifier incorrectly generalizes into unmeaningful user action region. The two highlighted areas at the left and the right sides in Figure 13 are the rejection regions of the second type.

We could address the second type of rejection by calculating the data likelihood from the trained classifiers, and asserting that the likelihood is above some pre-defined threshold. Particularly, let $\vec{x}$ denote the sensor data, and $p(\vec{x})$ denote the likelihood of $\vec{x}$. As for the NB classifier, we can calculate the likelihood by

$$\begin{aligned} p(\vec{x}) &= \sum_{c \in C} p(\vec{x}, c) \\ &= \sum_{c \in C} p(\vec{x}|c)p(c) \\ &\approx \sum_{c \in C} p_{NB}(\vec{x}|c)p_{NB}(c), \end{aligned}$$

where $p_{NB}(\vec{x}|c)$ is the likelihood calculated from the trained NB classifier,

$$p_{NB}(\vec{x}|c) \sim N(\vec{\mu}_c, \vec{\sigma}_c)$$

and $p_{NB}(c) = 1/|C|$ is the prior on the classes. We set a threshold τ so that the classifier produces classification results only when $p(\vec{x}) >= \tau$. Unfortunately, this simple calculation scheme *alone* does not yield useful results.

Data from Class	AVG	SD
CALL	-347.44	18.59
SMS (one-handed)	-348.52	20.15
SMS (two-handed)	-348.81	19.77
CAMERA (horizontal, one-handed)	-342.37	22.55
CAMERA (vertical, one-handed)	-341.75	19.46
CAMERA (two-handed)	-348.14	18.88
VIDEO	-351.94	20.33
GAME	-353.47	19.79
no touch	-345.00	21.69

Figure 14: Averages (AVG) and standard deviations (SD) of log likelihoods $\log(p_{NB}(\vec{x}))$ of training data

Figure 14 shows the averages and standard deviations of log likelihoods for training instances of each class. The values are calculated from the trained naive Bayes classifier. When the prototype device is put on a table and not touched at all, the average log likelihood is -345.00 with the standard deviation of 21.69. Our hope was that the log likelihoods of no-touch data will be substantially smaller than those of training data, but it was not the case. If we use the simple rejection scheme based on the data likelihood, we will be rejecting a lot of intentional and legitimate hand grips.

We then moved on experimenting with recent developments in the SVM research in determining whether a given data comes from an underlying probability distribution. This type of SVM is called the called one-class SVM (Schölkopf *et al.* 2000). Specifically, the one-class SVM is designed

ν	Rejection Ratio
0.05	0.049
0.10	0.099
0.13	0.129
0.15	0.150
0.2	0.200
0.3	0.301
0.5	0.501

Figure 15: Experiments with one-class SVM (normalization preprocessing and linear kernel)

to handle problems where only positive examples are available for the training, *i.e.*, one-class problems. The one-class problem setting particularly suits our situation since it is almost impossible to ask subjects to hold the prototype device *unintentionally*. The one-class SVM has been applied to domains such as document classification (Manevitz & Yousef 2001), machinery monitoring (Unnthorsson, Runarsson, & Jonsson 2003), and computer security (Wang & Stolfo 2003). We adjusted the parameter ν of the one-class SVM so that it rejects a small portion of the training set. Figure 15 shows the value of ν and the rejection ratio of the training data. We are currently experimenting with $\nu = 0.3$, rejecting almost 30% of the training data. Preliminary user evaluation indicates that this value is optimal.

Conclusions and Future Work

In this paper, we described a sensor-based mobile user interface employing an array of capacitive touch sensors and a 3-axis accelerometer in order to identify the hand grip patterns of the mobile handheld device and direct the user to the intended mobile application.

We developed a working prototype device in order to verify the feasibility of the idea, and showed that a naive application of traditional pattern recognition technique to classify the grip pattern is not acceptable. We identified various engineering issues, such as the limited available memory of a typical handheld device and the rejection of unintentional skin touch for reliable operation.

Currently, we are looking into the interpretations of various classifier parameters in terms of *perceived* classification accuracy and user satisfaction score. Ideally, we will be able to define a function that maps from the classifier parameter values to the usability scores.

We are also investigating the applicability of support vector representation and discrimination machine (Yuan & Casasent 2003) for unifying the rejection scheme and the classification scheme into a single framework. Addressing the on-line learnability of the grip pattern classifier is also an important problem for the future work. The on-line learning algorithms would be able to make the system adaptable, in the sense that as the user interacts with the system by giving classification error feedback, the system will adapt to the *personalized* hand grip patterns of the user. We are also experimenting with on-line reinforcement learning algorithms (Kaelbling, Littman, & Moore 1996) to address the issue of adaptability.

References

Chow, C. K. 1970. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory* 16(1).

Duda, R.; Hart, P.; and Stork, D. 2001. *Pattern Classification*. John Wiley & Sons, Inc.

Hinckley, K.; Pierce, J.; Sinclair, M.; and Horvitz, E. 2000. Sensing techniques for mobile interaction. In *Proceedings of Symposium on User Interface Software & Technology*.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. P. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4.

Manevitz, L. M., and Yousef, M. 2001. One-class svms for document classification. *Journal of Machine Learning Research*.

Norman, D. A. 1988. *The Psychology of Everyday Things*. Basic Books.

Platt, J. 1998. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*.

Poupyrev, I.; Maruyama, S.; and Rekimoto, J. 2002. Ambient touch: Designing tactile interfaces for handheld devices. In *Proceedings of Symposium on User Interface Software & Technology*.

Rekimoto, J. 2002. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems*.

Schölkopf, B.; Platt, J.; Shawe-Taylor, J.; Smola, A. J.; and Williamson, R. C. 2000. Estimating the support of a high-dimensional distribution. In *Advances in Neural Information Processing Systems*.

Shen, C.; Vernier, F. D.; Forlines, C.; and Ringel, M. 2004. Diamondspin: an extensible toolkit for around-the-table interaction. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems*.

Unnthorsson, R.; Runarsson, T. P.; and Jonsson, M. T. 2003. Model selection in one-class ν-svms using rbf kernels. In *Proceedings of Conference on Condition Monitoring and Diagnostic Management*.

Wang, K., and Stolfo, S. J. 2003. One class training for masquerade detection. In *Proceedings of ICDM Workshop on Data Mining for Computer Security*.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

Yuan, C., and Casasent, D. 2003. A novel support vector classifier with better rejection performance. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.