Multi-Touch Haptic Gesture Recognition from Examples

Nellie Margaret S. Chua[1], Demi Rose P. Tinay[2], Danielle Erika Y. Ureta[3],
Angela Faye C. Chua[4], Solomon L. See[5]
College of Computer Studies
De La Salle University
2401 Taft Avenue, Manila, Philippines
nelliechua526@yahoo.com[1], drtinay@yahoo.com[2], densagara@yahoo.com[3],
gelachua@yahoo.com[4], solomon.see@gmail.com[5]

ABSTRACT
At present, developers hand code their touch screen applications to recognize and react to unconventional touch gestures for special needs. In this paper, we present a Hidden Markov Model toolkit for gesture recognition designed to save developers time and effort in developing gesture-based applications for handheld devices.

General Terms
Algorithms, Human Factors

Keywords
User Interface Toolkits, Touch Screens, Hidden Markov Models, Machine Learning

1. INTRODUCTION
Gestural interaction has been identified as an important component of modern multi-modal interfaces because a collection of simple 2D gestures can represent myriads of various actions. Developers at present are required to hand code haptic gesture recognition into their touch screen applications, making development more difficult since various factors, including but not limited to finger offsets, incidence angles, etc. have to be considered. The toolkit to be described herewith was designed to recognize gestures from examples and integrate trained gestures into touch screen applications for handheld devices.

This paper will include a discussion of related works, the machine learning technique utilized in our design, an initial prototype, as well as results from experiments done on the prototype. The gesture recognition engine was designed to run on the Apple iPhone and to recognize both single-stroke and multi-touch gestures. A multi-touch haptic gesture is defined in this study as a continuous finger-based touch gesture that begins with the onset of the first finger on the touch surface and ends with the lift-off of all fingers. As illustrated in Figure 1, the engine developed can differentiate such gestures in terms of position (i.e. a gesture made at the top of the touch surface is different from a similar gesture made at the bottom of the surface), rotation (i.e. a top-to-bottom "X" gesture is different from a left-to-right "X" gesture), and scale (i.e. a big "X" gesture across the touch surface is not the same as a small "X" gesture at the center of the display).

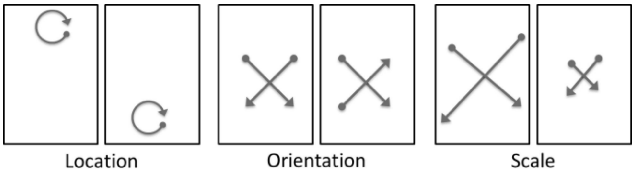

Figure 1 Different Locations, Orientations, and Scales

2. RELATED WORK
2.1 Machine Learning (ML) Algorithms
The most common algorithms used in the recognition of various modes of human-computer interaction are Artificial Neural Networks (ANN), Support Vector Machines (SVM), and Hidden Markov Models (HMM). These algorithms are often chosen for recognition systems primarily because of their learning capability and accuracy. These algorithms are dependent on the quality of extracted motion features which reflect the non-linear nature of human motions [6].

2.1.1 Artificial Neural Networks (ANN)
ANNs are mathematical models inspired by biological neural networks whose information processing uses a connectionist approach. Different ANN algorithms have been used in the study of recognition systems. Some examples are (1) Murakami and Taguchi's recurrent neural network for recognizing symbols from Japanese sign language gestures [15], and (2) Modler and Myatt's time-delay neural networks (TDNN) for recognizing hand gestures as patterns. As such, both studies utilized neural networks since neural networks are known for pattern recognition. Though the two studies implemented different ANN algorithms, both still rely on the basic principles of using multiple inputs for sampling to achieve a set of user-defined desired outputs. Both studies observed recognition rates that imply better accuracy given more training patterns.

2.1.2 Support Vector Machines (SVM)
SVMs are supervised, non-probabilistic binary linear classifiers. SVM algorithms are mainly used in human recognition systems (e.g. face recognition and detection). Li, et al. studied multi-view face detection and recognition using SVM and claims that SVMs address the problem of rotation out of the image plane that ANN-based systems have not [12]. They showed that using their system on ten subjects, implementing SVM yielded 93% recognition accuracy. A recent study by Dongwei, et al. cites multiple reasons for their choice of SVMs:

(1) Effectiveness in terms of image classification as proven in a study by Schölkopf, et al.
(2) Simplicity of process in extracting the set of data for classification, which is done automatically or implicitly through SVM
(3) School of thought is similar to ANN, but addresses a limitation of ANNs. SVMs optimally divide a data set into two categories for classification and make use of the one-versus-the-rest strategy [6], which provides room for flexibility in terms of recognizing unknown or undefined data.

2.1.3 Hidden Markov Models (HMM)

HMMs were originally developed for speech recognition, but have later become popular in researches concerning gesture recognition. Chen et al. conducted a research for hand gesture recognition involving an HMM using a real-time tracking method. They collected 60 (20 people, 3 times each) different image sequences for each of 20 pre-defined gestures and used these for their data. Their system made use of Fourier descriptors to extract hand gestures, enabling characterization of spatial features and motion analysis on the videos. Their experiment yielded a recognition rate of 97% using the training data for testing and 90.5% using separate testing data, which increased to 98.5% and 93.5% respectively when motion vectors were added to the use of Fourier descriptors. Another study by Webel, et al. cites that HMM provides a probabilistic framework that can account for time-varying and dynamic gestures [22]. This study covers multi-touch gestural interaction on a table-top display where gestures by both experienced and novice users can be made. The training phase of the study involved eight individuals performing a gesture 20 times to account for possible differences. The research yielded 97.5% recognition accuracy.

Table 1. Table of Comparison for ML Algorithms

Criterion	ANN	SVM	HMM
Technique	Multiple inputs mapped to a single user defined output	Multiple inputs mapped to a single user-defined output but divides data set into two categories for classification	Uses probability to output models and takes into consideration time-varying and dynamic inputs
Adaptability	Outcome is always fixed (no room for dynamic variations)	Flexible (undefined data can be accepted if similar to defined data)	Very Flexible (Data is de-fined based on probabi-listic inputs)
Recognition Accuracy	71.4 – 98%	90 – 95%	90.5 – 98.5%
Common Applications	Gesture or motion recognition (mostly video-based inputs)	Face recognition and detection (also covers motion recognition)	Originally developed for speech recognition (now widely used for gesture recognition researches)

HMM was chosen for this study primarily due to its recognition accuracy as seen in relevant previous studies as well as flexibility in terms of application as can be seen in Table 1.

2.2 Related Systems and Toolkits

Carnegie Mellon University's Landay and Myers have extended the *Garnet User Interface Development Environment* to support pen-based and mouse-input gestures in applications for desktop workstations [10]. They created Garnet gesture interactors from classifiers made based on distinguishing features extracted statistically from examples. Their study concluded that gestures should be considered as a basic input type and suggested improvements in terms of context-dependency as well as explicit declaration of gestures being size- and/or orientation-independent.

The *Gesture Learning and Recognition System* by Damaraju and Kerne has a learning stage which makes use of *k*-means clustering and HMM (Baum-Welch algorithm) to learn hand gestures by example [8]. It also has a recognition stage for which 40 different gestures were defined to demonstrate and test the subtleties of finger movements. The said system averaged a recognition rate of 91.5% on large multi-touch screens which makes it reasonable for practical use. Similarly, Webel et al. made a gesture recognition module with a filtering phase prior to a training/learning phase employing *k*-means clustering and HMM, and a recognition phase. Training data were collected on a touch table and testing was done on both a computer's desktop (via mouse) and the touch table. The accuracy of recognition of this study are relatively higher than those of Damaraju and Kerne's system.

Another interesting work is the *Stroke Shortcuts Toolkit* by Appert and Zhai [2]. This toolkit is a Java Swing extension to simplify addition of stroke shortcuts to Swing applications. It has a structured yet open environment where designers and/or developers define strokes through a design space. New strokes can be defined simply by transforming and/or combining strokes in a predefined dictionary. This dictionary eliminates the need for a training phase for its recognition engine, but it also limits the number of possible gestures to an extent. The strokes are collected from pen- and mouse-based inputs, and the error rates of the system for both are relatively insignificant and almost negligible.

The following table provides a summary and a comparison of these systems.

Table 2. Table of Comparison for Related Systems and Toolkits

System	Input	Technique	Limitation
Landay and Myers	Pen and mouse	Statistical extraction of distinguishing features	Context-dependency, independence from size and orientation
Damaraju and Kerne	Multi-touch hand	K-means clustering, HMM, Baum-Welch algorithm	Tested on large multi-touch displays only, data/ feature set in video format
Webel	Hand and mouse	K-means clustering, HMM	Tested on touch tables (hand) and computer desktops (mouse)
Appert and Zhai	Pen and mouse	Modification of pre-defined strokes	Strokes limited to pre-defined dictionary

3. THE MULTI-TOUCH HAPTIC GESTURE RECOGNITION TOOLKIT

The toolkit we implemented reduces the burden of developers via abstracting the process of hand-coding gesture recognition. It provides developers an Application Programming Interface (API) and other files helpful for gesture recognition, resulting to less complex and fewer lines of code. The toolkit consists of two main applications: (1) The data collection application installed in the handheld device (Apple iPhone in this study) and (2) The desktop toolkit. The handheld application serves as the medium for gesture inputs, for collecting sample data, and connecting to the desktop toolkit, which in turn is a platform for training and model generation. Specifically, the desktop toolkit runs the training and preliminary testing phases of the machine learning algorithm (HMM).

3.1 Architectural Design

The toolkit has three modules: Collection, Training, and Recognition. The Collection Module is where the system collects and records sample gesture data. This module will run on the touch screen device (iPhone). Collected gesture data are then transferred into a gesture library/database in a desktop computer, where the Training and Recognition modules are run. With this setup, the user can store numerous amounts of gesture data and has the liberty to choose which types of gestures they intend to use for their applications, without being bound by the iPhone's memory and processing limitations (that is, opposed to a PC). User-selected gesture data will then serve as training examples for the Training Module. The Training Module, which is made up primarily of a user interface with background HMM processes, creates the corresponding gesture model to be fed to the Recognition Module, which tests the accuracy of gesture recognition via a naïve Bayesian classifier. Once a satisfactory recognition accuracy (relative to the user's objective measure) is met, the user can opt to use the trained HMM gesture models as input to the API file or object for a recognition application in the iPhone which will perform recognition when a touch event is passed onto it.

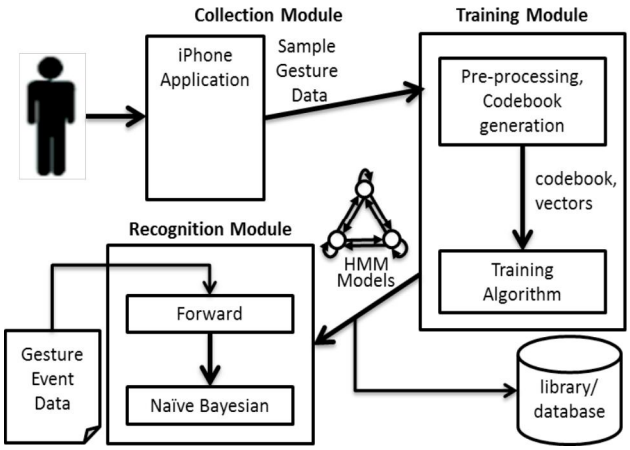

Figure 2 Architectural Design of the Toolkit

3.2 Collection Module

This module aids in collecting data from sample gestures. Its main component is a Gesture Input Application (GIA) that stores into a text file a series of Cartesian coordinates corresponding to the touch points collected at a rate of 60 sets per second. These text files can then be transferred to the training module (in the desktop toolkit application) via SyncDocs, a GNU-licensed application for transferring files through the Bonjour discovery protocol of Apple.

For this study, sample gesture data were collected from 20 randomly chosen individuals, some of whom are experienced iPhone users. Each user provided at least 5 samples of each of the gestures used in the study.

3.3 Training Module

3.3.1 Data Preprocessing

Prior to training, collected gesture data must be pre-processed into observation vectors. This can be done through feature extraction and vector quantization.

Feature selection and extraction are used to reduce bandwidth of input data, to remove redundant or irrelevant information, or to provide a relevant set of features for the classifier [21]. The features most used for gesture recognition are templates, global transformations, zones, and geometric features [23]. For this study, the set of features utilized includes: (1) the set of point coordinates, (2) the location, (3) the direction, and (4) the velocity at which the gesture is performed. These are done via converting to polar coordinates (in the form $(\sqrt{(\Delta x)^2 + (\Delta y)^2}, arctan(\frac{\Delta y}{\Delta x}))$ the transition between every pair of corresponding Cartesian coordinates, taking note of the first point of contact, and collecting points at a uniform rate. The generated polar coordinates are then converted into finite symbols through vector quantization (VQ).

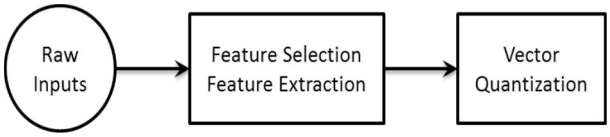

Figure 3 Preprocessing of raw inputs for training

VQ is the process of encoding of a p-dimensional vector x as a value from a set *codebook* of g vectors, $z_1,\ldots, z_g$, termed as the *code vectors* or the *codewords* [21]. In this study, k-means clustering is used to generate the codebook, ensuring the code vectors to be specific to the set of sample gestures intended for training and recognition. Once the codebook is complete, separate text files are created to store the corresponding code vectors for the pre-processed gesture data.

3.3.2 HMM Training

HMM is a doubly stochastic process involving sequences of states (which are hidden and internal to the model) and of emission symbols (which are observable) [9]. Formally, an HMM is defined with the following parameters:

(1) N : number of states of the model
(2) M : number of observed states
(3) Π : an array of initial state probabilities
(4) A : an $N \times N$ matrix of state-to-state transitional probabilities
(5) B : an $N \times M$ matrix of state output probabilities

In this study, two approaches to HMM training have been reviewed, implemented, and analyzed. The first one makes utilizes the well-known Baum-Welch algorithm and an ergodic (fully connected, i.e. a state can transition to all other states) topology. The second one uses an optimized/modified algorithm by Vesa-Matti Mantyla [14] and a left-right (strictly uni-directional connection, a state can only transition to a subset of the states, which may include itself, but generally following one direction) topology.

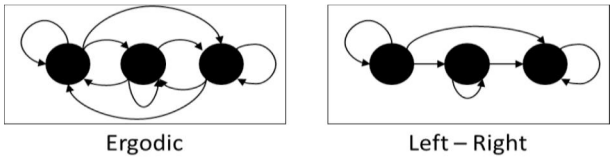

Figure 4 Ergodic vs Left-Right Topology

The Baum-Welch algorithm is an Expectation Maximization (EM) algorithm for re-estimating the three probabilities (Π, A, B) of an HMM model. It is a process that iterates the following steps until a local maximum (has been proven good enough so it is used instead of the global maximum, computing for which is intractable) is reached or a pre-defined number of iterations is met:

(1) Selection of initial values for λ (Π, A, B)
(2) Determination of probable sequences of states
(3) Computation of the expected number of each of the state-to-state transition (S)
(4) Computation of the expected number of times each symbol is emitted from a state (E)
(5) Re-estimation of λ from S and E.
(6) If not converged, repeat from step 2.

Mantyla's modified algorithm involves the same steps as the Baum-Welch algorithm, excluding step 6. Reiteration is no longer necessary because of slight re-estimation formula modifications that are anchored on certain assumptions and observations applicable to the left-right topology.

Both algorithms employ (for determination of probable state sequences) the Forward-Backward algorithm, a smoothing procedure for HMMs which involves three steps:

(1) Computing for forward probabilities (probability of transitioning to a particular state given the first k observations in a sequence)
(2) Computing for backward probabilities (probability of observing the remaining observations given a point k)
(3) Combining the forward and backward probabilities to compute for smoothed values.

3.4 Recognition Module

The recognition module is an intermediate step between the training and the deployment of the HMM models. It provides users the facility to test the accuracy of the Training Module's output HMMs. Recognition is done through a naïve Bayesian classifier given the respective probabilities of the models involved.

4. RESULTS AND DISCUSSION

Two ways have been designed to test the gesture recognition accuracy. One is through the Recognition Module described above, and another is through a prototypic iPhone casual cooking game which utilizes the system's API containing the trained models for the gestures designed for the application.

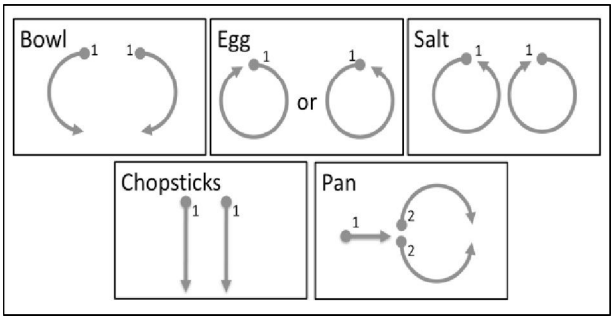

Figure 5 Multi-touch gestures required for the game

The game involves performing the gestures illustrated in Figure 5 as instructed by the player's chosen recipe, either fried egg or omelet. When the user performs a gesture incorrectly, the system will display a red "X" on screen, and will only proceed to animation and the succeeding instruction once the user is able to perform the gestures correctly. The game ends when the user successfully performs all the gestures necessary for his/her chosen recipe.

The two training algorithms described in the previous section have been tested. These two algorithms have been compared based on speed and accuracy of both training and recognition. Table 3 is a summary of the results.

Table 3. Comparison of HMM Algorithms

	Ergodic – Baum Welch	Left-Right – Modified Algorithm
Training Speed	6 to 30 hours	4 to 620 seconds
Training Accuracy	73%	98%
Recognition Speed	4 seconds	40 milliseconds
Recognition Accuracy	62%	80%

For each algorithm, Training Speed is shown as a range of the mean running times for training different sets of gestures on desktop computers while Recognition Speed is the mean running time of recognition for a specific gesture on the iPhone. Training Accuracy shown is a mean of the accuracy of recognition (number of correctly identified gestures over total number of gestures fed) of gestures chosen from the set of training data while Recognition Accuracy is a mean of the accuracy of recognition of real-time on-device (iPhone) gestures outside the training set. As can be seen from Table 3, the left-right model using the modified algorithm in [14] has a significantly better performance.

Various tests have also been run to gauge the accuracy of models generated under different circumstances. A summary of the results are as follows:

Table 4. Comparison of Accuracy Given Number of States

Gesture	Different			Constant		
	Failed	Misre-cognized	Accurate	Failed	Misre-cognized	Accurate
Bowl	0%	24%	76%	0%	22%	78%
Chopsticks	0%	18%	82%	0%	18%	82%
Egg	0%	32%	68%	0%	0%	100%
Pan	70%	0%	30%	0%	0%	100%
Salt	27%	3%	70%	40%	0%	60%

One HMM model per gesture is generated per training run. Table 4 shows a summary of the results comparing accuracy of recognition given that training was done with (1) different number of hidden HMM states for each gesture in the set and (2) constant number of hidden HMM states for all gestures in the set. It can be seen that a constant number of hidden states for all the gestures may be more favorable.

Table 5. Recognition Accuracy Given Samples with Noise

Gesture	Failed	Misrecognized	Accurate
Bowl	5%	30%	65%
Chopsticks	7%	15%	78%
Egg	0%	11%	89%
Pan	29%	0%	71%
Salt	15%	27%	48%

The training module comes with a visualization panel where users can choose a file containing the points corresponding to an instance of a gesture. Noisy samples are defined in this study as those gestures whose visualizations are too far away from the respective intended gesture. Some of the first-time touch screen device users who provided gesture samples for the study have contributed a number of noisy samples (e.g. unfinished gestures, incorrect finger incidence angles which contributes to faulty point determination on the device (outside the scope of this study), improperly positioned gestures, etc.). Table 5 and Table 6 show the

same set of gestures with and without noisy samples, respectively. Although the system was expected to tolerate noisy data which may be brought about by dissimilarities among the users, it can be observed that performance is slightly better for training data without noise.

Table 6. Recognition Accuracy Given Samples without Noise

Gesture	Failed	Misrecognized	Accurate
Bowl	0%	33%	67%
Chopsticks	0%	1%	99%
Egg	0%	3%	97%
Pan	1%	18%	81%
Salt	8%	40%	52%

Four sets of ambiguous gestures have also been tested on the system. Three of the four sets are as follows:

(1) Check mark ("✓") and the 22nd letter in the English alphabet ("V")
(2) The Arabic numeral for five ("5") and the English alphabet's 19th letter ("S")
(3) The Arabic numeral for two ("2") and the last letter in the English alphabet ("Z")

Tests ran using these gesture sets generated results which are similar to those previously shown. Figure 6 illustrates the more significant set of ambiguous gestures, "9", "g", and "q", along with a visualization of corresponding sample user input for each.

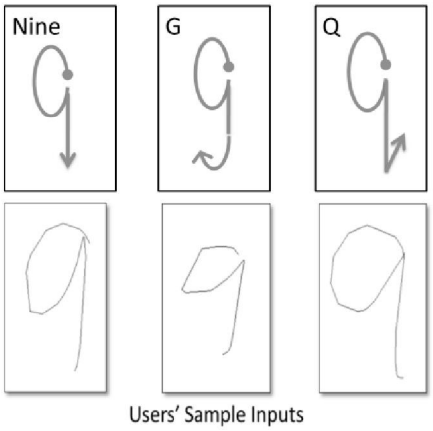

Figure 6 Ambiguous Gestures and Corresponding User Input

Table 7 shows the system's performance given the ambiguous gestures illustrated above. It should be noted that this performance is significantly worse than those of non-ambiguous gestures (as are illustrated in Figures 4, 5, and 6). However, further study is necessary to make a sound conclusion.

Table 7. Recognition Accuracy Given Ambiguous Gestures

Gesture	Failed	Misrecognized	Accurate
g	0%	2%	98%
q	0%	100%	0%
9	0%	20%	80%

5. CONCLUSION

Based on the results of the study, the proponents would like to conclude the feasibility of using HMM for real-time and practical multi-touch gesture recognition. Some observations for achieving better performance of the system include:

(1) Given the sample-based codebook generation, training the set of gestures to be distinguished from each other are better trained together.
(2) Training roughly the same amount of samples for similar gestures to be distinguished from each other improves system performance.
(3) Minimization of noise in samples makes room for better recognition accuracy and less required samples.
(4) Assigning constant values for the number of clusters (code vectors) and number of states for all models of gestures in a set can slightly improve recognition accuracy.

6. FUTURE WORK

Further study using our system can focus on finding the ideal values for the number of clusters (i.e. size of the codebook) and number of samples. Other studies can also be done on similar systems using other clustering or quantization algorithms, learning algorithms, and classifiers (for recognition). Future work can also consider the following: (1) further analysis for the ambiguous gestures, (2) support for non-continuous gestures (e.g. taps, double taps, etc.), (3) the incorporation of the recognition process into the operating system of the handheld device, so that creation of custom gestures is more accessible to both developers and handheld touch screen device users.

7. ACKNOWLEDGMENTS

This work was done under the Learning Innovation Center of the College of Computer Studies, De La Salle University (DLSU). Our most sincere gratitude goes to research groups sharing valuable resources and tools, to faculty and personnel who extended their utmost support and assistance, as well as to DLSU for providing iMacs and an iPhone unit for our development and testing.

8. REFERENCES

[1] Albinsson, P.-A., and Zhai, S. (2003). High precision touch screen interaction. *Chi '03: Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 105{112). New York, NY, USA: ACM.

[2] Appert, C., and Zhai, S. (2009). Using strokes as command shortcuts: cognitive benefits and toolkit support. In *Chi '09: Proceedings of the 27th international conference on human factors in computing systems* (pp. 2289{2298). New York, NY, USA: ACM.

[3] Apple, I. (2009a). *Event handling. iphone programming guide*. Author Available from http://developer.apple.com/iphone/library/navigation/index.html

[4] Apple, I. (2009b). *The objective-c 2.0 programming language*. Retrieved July 10, 2009, from http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/ObjectiveC/ObjC.pdf

[5] Brooks, A. (2009). *Apple awarded multi-touch patent*. Retrieved July 10, 2009, from http://news.worldofapple.com/archives/2009/01/26/apple-awarded-multi-touch-patent/

[6] Cao, D., Masoud, O. T., Boley, D., and Papanikolopoulos, N. (2009). Human motion recognition using support vector machines. *Computer Vision and Image Understanding*, *In Press, Corrected Proof* , -. Available from http://www.sciencedirect.com/science/article/B6WCX-4WK48F7-1/2/2305c6de1dacbbf4383625a5aea41d74

[7] Chen, F.-S., Fu, C.-M., and Huang, C.-L. (2003). Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Vision Computing*, *21* (8), 745 - 758. Available from http://www.sciencedirect.com/science/article/B6V09-48NC790-1/2/8f12375d2a82de6de563284fd02d3f23

[8] Damaraju, S., and Kerne, A. (1998, Oct.). Multitouch gesture learning and recognition system. In *Extended abstracts of ieee workshop on tabletops and interactive surfaces*.

[9] Kolesnik, P. (2004). *Conducting gesture recognition, analysis and performance system*. Unpublished master's thesis, McGill University, McGill University, Montreal, Canada.

[10] Landay, J. A., and Myers, B. A. (1993). Extending an existing user interface toolkit to support gesture recognition. In *Chi '93: Interact '93 and chi '93 conference companion on human factors in computing systems* (pp. 91{92). New York, NY, USA: ACM.

[11] Lane, S. (2007, August). *Apple developing configurable multi-touch gesture dictionary*. Retrieved July 10, 2009, from http://www.appleinsider.com/articles/07/08/02/apple developing configurable multi touch gesture dictionary.html

[12] Li, Y., Gong, S., Sherrah, J., and Liddell, H. (2004). Support vector machine based multi-view face detection and recognition. *Image and Vision Computing*, *22* (5), 413 - 427. Available from http://www.sciencedirect.com/science/article/B6V09-4BP9P9W-3/2/737b1eabb29ce570bf189b7f199b4f87

[13] Luk, J., Pasquero, J., Little, S., MacLean, K., Levesque, V., and Hayward, V. (2006). A role for haptics in mobile interaction: initial design using a handheld tactile display prototype. In *Chi '06: Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 171{180). New York, NY, USA: ACM.

[14] Mantyla, V.-M. (2001). *Discrete hidden markov models with application to isolated userdependent hand gesture recognition* (VTT publication No. 449). Espoo, Finland: VTT. Retrieved April 22, 2009 from http://www.vtt.fi/inf/pdf/publications/2001/P449.pdf

Murakami, K., and Taguchi, H. (1991). Gesture recognition using recurrent neural networks. In *Chi '91: Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 237{242). New York, NY, USA: ACM.

[15] Nacenta, M. A., Baudisch, P., Benko, H., and Wilson, A. (2009). Separability of spatial manipulations in multi-touch interfaces. In *Gi '09: Proceedings of graphics interface 2009* (pp. 175{182). Toronto, Ont., Canada, Canada: Canadian Information Processing Society.

[16] Roudaut, A. (2009). Visualization and interaction techniques for mobile devices. In *Chi ea '09: Proceedings of the 27th international conference extended abstracts on human factors in computing systems* (pp. 3153{3156). New York, NY, USA: ACM.

[17] Roudaut, A., Huot, S., and Lecolinet, E. (2008). Taptap and magstick: improving one-handed target acquisition on small touch-screens. In *Avi '08: Proceedings of the working conference on advanced visual interfaces* (pp. 146{153). New York, NY, USA: ACM.

[18] Stengel, M. (2003, March). Introduction to graphical models, hidden markov models and bayesian networks. Toyohashi, Japan: Toyohashi University of Technology.

[19] Tanguay, D., Jr. (1995). Hidden markov models for gesture recognition. In (pp. 1{52). Massachusetts Institute of Technology, Cambridge, MA: Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[20] Webb, A. (2002). Statistical pattern recognition. In (Second ed., pp. 305{360). The Atrium, Southern Gate, Chichesterm West Sussex P019 8SQ, England: John Wiley and Sons Ltd.

[21] Webel, S., Keil, J., and Zoellner, M. (2008). Multi-touch gestural interaction in x3d using hidden markov models. In *Vrst '08: Proceedings of the 2008 acm symposium on virtual reality software and technology* (pp. 263{264). New York, NY, USA: ACM.

[22] Yang, J., and Xu, Y. (1994, May). *Hidden markov model for gesture recognition* (Tech. Rep. No. CMU-RI-TR-94-10). Pittsburgh, PA: Robotics Institute