GestureCommander: Continuous Touch-based Gesture Prediction

George Lucchese
Department of Computer Science and Engineering
Texas A&M University
george_lucchese@tamu.edu

Martin Field
Department of Computer Science and Engineering
Texas A&M University
martin.field@gmail.com

Jimmy Ho
Department of Computer Science and Engineering
Texas A&M University
jimmyho@tamu.edu

Ricardo Gutierrez-Osuna
Department of Computer Science and Engineering
Texas A&M University
rgutier@cse.tamu.edu

Tracy Hammond
Department of Computer Science and Engineering
Texas A&M University
hammond@cs.tamu.edu

CHI'12, May 5-10, 2012, Austin, Texas, USA.
ACM 978-1-4503-1016-1/12/05.

Abstract
GestureCommander is a touch-based gesture control system for mobile devices that is able to recognize gestures as they are being performed. Continuous recognition allows the system to provide visual feedback to the user and to anticipate user commands to possibly decrease perceived response time. To achieve this goal we employ two Hidden Markov Model (HMM) systems, one for recognition and another for generating visual feedback. We analyze a set of geometric features used in other gesture recognition systems and determine a subset that works best for HMMs. Finally we demonstrate the practicality of our recognition HMMs in a proof of concept mobile application for Google's Android mobile platform that has a recognition accuracy rate of 96% over 15 distinct gestures.

Author Keywords
Gesture Recognition; Mobile Computing

ACM Classification Keywords
I.5.5 [Pattern Recognition]: Implementations—*Interactive Systems*

General Terms
Human Factors

Introduction

Many applications use sketched gestures as a form of command input, especially on mobile platforms, where multi-touch technology is becoming ubiquitous. By executing a specific gesture, the user is able to easily invoke "hidden" functionality that does not require any visible affordances. A command gesture can be considered similarly to a keyboard shortcut; however, unlike a keyboard, there is ambiguity when trying to interpret a users sketched input.

Many approaches to gesture recognition have been studied: template matching [12], feature-based [11, 9], and signal-based [1]. Template matching approaches are often very simple and easy to implement, but their run-time increases with the size of the training data. Template matching is often used for interface prototypes, or in situations where quick retraining of the recognizer is necessary. Feature-based and signal-based approaches are closely related, and both have the drawback of requiring large training data sets. These approaches are often used when recognition speed must be fast and computational resources are low, since most of the work can be done offline during the training phase.

One special class of gesture recognizers is able to begin classifying the gesture while the user is still in the process of drawing. This type of recognizer is often used for interactive gesture-based menu systems [6]. The continuous recognition allows the user to perform multiple actions, such as navigate a hierarchical menu or adjust parameters, without lifting the pen. Some systems, such as OctoPocus [5] and Flower Menus [4], can display a visualization of the current recognition status, including all of the available continuations of the gesture in progress. These systems demonstrate that in-progress visualization is an effective tool to help novice users discover what gestures are offered and what they do.

This paper proposes a gesture recognition, prediction, and feed-forward system using an HMM-based recognizer and gesture extrapolation engine for use on touch-based mobile device interfaces. The system is able to expose the state of recognition as the user is performing a gesture, allowing for mid gesture correction or cancellation. Additionally, since recognition results are available continuously, the system is able to anticipate user commands before they are completed, allowing for background actions such as caching to be performed, and potentially decreasing perceived response times for the user. As part of this system, we describe a set of geometric features that we extract from gestures that we found to be practical for training continuous Hidden Markov Models (HMM). We also demonstrate the practicality of our recognition HMMs on a proof of concept mobile application for the Android platform.

Related Work

Gesture Recognition

There are three main approaches to two-dimensional gesture recognition: template matching, feature based recognition and signal-based recognition. For template matching, one of the best known approaches is the $1 recognizer of Wobbrock et al. [12]. $1 uses a set of predefined template gestures against which all user entered gestures are compared for recognition. Both the templates and unknown gestures are resampled to a fixed inter-point distance, are scaled and rotated, and are then compared on a point-by-point basis. This requires that the gesture be completed before recognition can occur. Other systems use techniques similar to $1 with modifications for multi-stroke gestures [2] and for

increased speed and accuracy [7].

Feature-based recognition uses the inherent geometry of gestures to compute a feature vector that may be used with traditional pattern analysis techniques. Rubine defines several useful features [11], some of which are described later, that have been frequently reused and extended. Like template systems, feature based recognition systems like Rubine's typically perform recognition after the user has completed a gesture.

Finally, there are signal-based gesture recognition systems that turn time based features of a gesture, such as changes in direction, velocity, or curvature into a continuous signal for recognition using HMMs. Notable examples of this are Anderson's gesture recognition system [1] and Ou's gesture and drawing classifier [8]. Of interest to us is Anderson's which uses a single feature, a direction vector calculated using a sliding window, that is binned and used in conjunction with a discrete HMM. This method yielded relatively high accuracy, 96% over a set of 11 gestures.

Mobile Gesture Systems
There are other mobile systems that use gestures to allow users to conduct a variety of tasks. Palm Graffiti is one of the best known mobile gesture systems, allowing users to enter text using gesture symbols. Another application in the same vein as Graffiti is Gesture Search, which lets users search their device by drawing a series of free drawn uppercase characters [7]. There are also feedback based text entry gesture systems, such as Swype and 8pen, that provide visual feedback to help the user recall and ultimately learn the gesture for a certain word.

Feed-forward
OctoPocus [5] is a system that teaches gestures to users by showing them the possible continuations from what they have already drawn. The likelihood of each continuation is visually conveyed through the weight and opacity of the stroke used to display it. OctoPocus uses a template matching recognizer, which caused the original system to be scale dependent. While systems like $1 avoid scale dependence by resizing each gesture to a standard size, that would not work to recognize a small part of an incomplete gesture. With the addition of a curvature template, OctoPocus was extended to support scale-independent recognition [3].

Online Recognition

For each gesture we apply the resampling algorithm from the $1 recognizer to the gestures as the user draws them. Each time a point is added to the resampled set, we extract features from that point and run another iteration of the forward algorithm [10] to update the current recognition status. Since the forward algorithm operates over the observation sequence in order, it is simple to update iteratively as the observation sequence is built. We chose an inter-point distance of 50 pixels, which on a mobile device typically represents a distance of 1/5 to 1/3 of an inch. For feed forward, we run the Viterbi algorithm to determine the current state of a gesture and the most likely state sequence for each model. The Viterbi algorithm iterates over the observation sequence in order, and is updated iteratively as the user draws.

Feature Extraction
Each observation is composed of features based on a subset of the Rubine features that could be easily extracted from each point without knowing the full stroke ahead of time. We investigated the following features:

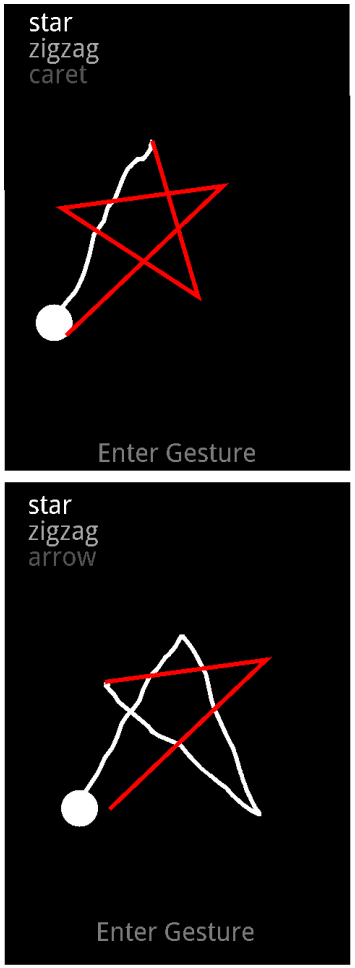

Figure 1: Screenshots from the GestureCommander application with drawn the gesture in white, gesture continuation in red and text recognition feedback.

direction, curvature, total curvature, angle between first and current point and bounding box diagonal angle.

Model Topology
As in [13], we used a left-to-right with no skips model. We further limit the model by forcing the model to start in the leftmost state and end in the rightmost state. We force the model to start in the leftmost state by initializing the pi matrix to have a 0 probability of starting in any other state. We force the model to end in the rightmost state by adding a special observation to the end of each sequence to represent the user lifting their finger.

GestureCommander Application

We demonstrate our gesture system with a simple proof-of-concept application called Gesture Commander, which runs on the Android platform (Figure 1). The system uses a modified version of the JaHmm library [1] to perform continuous recognition using the HMMs constructed in training. This application allows users to perform gestures while displaying textual feedback of what the system predicts to be the three most likely gestures, allowing users to modify or adjust the gesture if necessary. As a second form of visual feedback, the application also traces out the anticipated continuation of the most likely gesture using Viterbi calculations as described previously.

Results

Data Collection
To evaluate the recognition accuracy of the system we used a set of 15 gestures from the $1 gesture set [12], collected using a specialized data collection application for Android called SOUSAPhone [9][2]. We collected approximately 30 samples total for each gesture from four different users, for a total of 450 gesture samples. The original gesture set of $1 is 16 shapes, but through an oversight we only collected data for 15 of these, missing the X shape.

Feature Selection
Using 4-fold cross validation, we found that the direction feature alone performs at least as well as any other combination. The top performing combinations were (direction), (direction, sine of initial angle), and (direction, curvature, sine of initial angle). Based on the work of [13], which demonstrates how delta features can be used to help improve Markov generation, we chose the feature subset (direction, curvature, sine of initial angle), since curvature is similar to a delta feature.

Recognition Results
Using the optimal feature set of direction, curvature and sine of initial angle we performed 4-fold cross validation to evaluate the predicted recognition accuracy. The mean correctness was 96.5% across all gesture classes with standard deviation of 5.82 percentage points. Recognition was poorest for right square and left square shapes, with right square worst with an accuracy of 85.3%.

Discussion

Recognition
As show in Figure 2, the recognition rates for GestureCommander are reasonably high. The total rate of approximately 96% is comparable to the accuracy of Andersons discrete HMM recognizer (96%) on a similar data set and to Rubines feature based classifier (96%)

[1] JaHmm source code available at http://code.google.com/p/jahmm/

[2] Web- and Android-based clients available from http://srlweb.cs.tamu.edu/srlng/sousa/

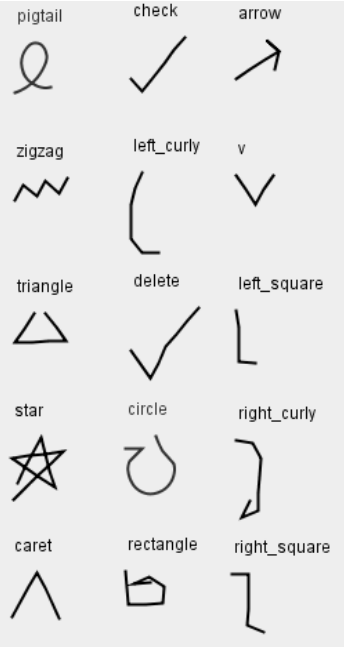

Figure 3: Shapes generated by the trained generator HMM

[12] on the same shape set. This is less accurate and consistent than the results of $1 based recognizers, which can get up to 99% accuracy on the same shape set, but $1 itself does not provide the continuous recognition that we need for gesture prediction and command anticipation.

Shape	Correct	Total	Accuracy
pigtail	29	29	100.0%
check	28	28	100.0%
arrow	30	30	100.0%
star	29	29	100.0%
circle	34	34	100.0%
zigzag	30	30	100.0%
left_curly	28	28	100.0%
v	29	32	96.87%
caret	26	28	92.86%
rectangle	28	28	100.0%
triangle	28	28	100.0%
delete	28	28	100.0%
right_curly	26	30	86.66%
left_square	24	28	85.71%
right_square	29	34	85.29%
			96.49

Figure 2: Recognition Accuracy Rates

Gesture Continuation

Using the HMM trained for classification we generated each gesture using the Viterbi algorithm to determine the most probable observation sequence. For each observation, we used the expected direction to generate a new point at a distance of 50 pixels away along that direction vector, the same distance used for resampling. As can be seen in Figure 3 this produces recognizable results which we use for our active gesture continuation in Gesture Commaner.

Future Work

The work in this paper is the first step in the development of a more complete HMM-based feed-forward gesture control system. To achieve this goal we must first expand GestureCommander into a practical and robust system that can be fully evaluated. We intend to implement a home screen launcher with the systems current visual feedback, feed-forward and continuous recognition. Users will be able to define and customize their own gestures, in addition to the predefined set included in the system, as well as assign applications or actions to them. Similar to the current implementation, the system will predict the top three gestures, display the gesture continuation and display the actions associated with the gesture.

With this completed we will be able to investigate the effectiveness of the system at providing user feedback and increasing perceived responsiveness of the system as a whole. In addition we hope to investigate practical issues of allowing users to define their own gestures, such as how to give feedback as to the separability of recognition classes and what other gestures a new gesture might be confused with by the recognition system.

Conclusion

We implemented GestureCommander, a signal-based continuous gesture recognition and feed-forward system. Using an HMM recognizer and three continuous features: direction, curvature and sine of initial angle, we were able to achieve recognition rates of 96% on a data set of 15 touch gestures. We showed that this recognition system is practical and usable on an example mobile system. We also showed that specially trained HMMs can produce reasonably accurate looking gesture continuations for user feed-forward purposes. Together, the continuous recognition and gesture generation allow us to provide

feedback and guidance to users as well as to anticipate their actions before completion.

Acknowledgements

This material is based upon work supported by Google and the National Science Foundation under grant numbers 0942400, 0935219, and 1129525. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors would like to thank members of the Sketch Recognition Lab for their continued support.

References

[1] Anderson, D., Bailey, C., and Skubi, M. Hidden markov model symbol recognition for sketch-based interfaces. In *AAAI Fall Symposium* (2004), 15–21.

[2] Anthony, L., and Wobbrock, J. A lightweight multistroke recognizer for user interface prototypes. In *Proc. of Graphics Interface 2010*, Canadian Information Processing Society, Canadian Information Processing Society (Toronto, Ont., Canada, 2010).

[3] Appert, C., and Bau, O. Scale detection for a priori gesture recognition. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, ACM (New York, NY, USA, 2010), 879–882.

[4] Bailly, G., Lecolint, E., and Nigay, L. Flower menus: a new type of marking menu with large menu breadth, within groups and efficient expert mode memorization. In *Proc. AVI '08 Proceedings of the working conference on Advanced visual interfaces* (2008), 15–22.

[5] Bau, O., and Mackay, W. Octopocus: a dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st annual ACM symposium on User interface software and technology* (2007), 37–46.

[6] Kurtenbach, G., and Buxton, W. The limits of expert performance using hierarchic marking menus. In *Proceedings of INTERACT93 and CHI93 conference on Human factors in computing systems* (1993), 482–487.

[7] Li, Y. Protractor: a fast and accurate gesture recognizer. *Proceedings of the 28th international conference on Human factors in computing systems* (2010), 2169–2172.

[8] Ou, J., and Chen, X. Gesture recognition for remote collaborative physical tasks using tablet PCs. *... in E-Learning and Collaboration* (2003).

[9] Paulson, B., and Hammond, T. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, IUI '08, ACM (New York, NY, USA, 2008), 1–10.

[10] Rabiner, L., and Juang, B. An introduction to hidden Markov models. *IEEE ASSP Magazine 3*, 1 (Jan. 1986), 4–16.

[11] Rubine, D. Specifying gestures by example. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques.*, SIGGRAPH, New York, NY, USA (1991), 329–337.

[12] Wobbrock, J., Wilson, A., and Li, Y. Gestures without libraries, toolkits or training: a 1 dollar recognizer for user inter- face prototypes. In *Proceedings of User Interface Software and Technology*, ACM (2007), 159–168.

[13] Wu, Y., and Wang, R. Minimum generation error training for HMM-based speech synthesis. In *2006 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006)* (2006).