A Recognition Safety Net: Bi-Level Thresholding for Mobile Motion Gestures

Matei Negulescu
University of Waterloo
David R. Cheriton School of Computer Science
mnegules@uwaterloo.ca

Jaime Ruiz
University of Waterloo
David R. Cheriton School of Computer Science
jgruiz@uwaterloo.ca

Edward Lank
University of Waterloo
David R. Cheriton School of Computer Science
lank@uwaterloo.ca

MobileHCI 2011, Aug 30–Sept 2, 2011, Stockholm, Sweden.
ACM 978-1-4503-0541-9/11/08-09.

Abstract
Designers of motion gestures for mobile devices face the difficult challenge of building a recognizer that can separate gestural input from motion noise. A threshold value is used to classify motion and effectively balances the rates of false positives and false negatives. We present a bi-level threshold recognizer that is built to lower the rate of recognition failures by accepting either a tightly thresholded gesture or two consecutive possible gestures recognized by a looser model. We evaluate bi-level thresholding with a pilot study in order to gauge its effectiveness as a recognition safety net for users who have difficulty activating a motion gesture. Lastly, we suggest the use of bi-level thresholding to scaffold learning of motion gestures.

Keywords
Bi-level thresholding, motion gestures, safety net

ACM Classification Keywords
H5.2. Information interfaces and presentation: User Interfaces (Interaction styles).

General Terms
Human Factors

Introduction

A unique challenge in mobile interaction design around motion gestures – gestures sensed by accelerometers that require a user to move the entire smartphone device in three dimensions – is the need to develop recognition algorithms that are sufficiently powerful to discriminate motion gestures from everyday device movement. Designers of recognizers (e.g. [4]) are faced with building systems that maximize the number of true activations and minimize rate of misrecognition. The typical technique to segment a gesture from an input stream is to create a threshold, i.e. a criterion value that represents a trade-off between false activations (false positives) and false non-activations (false negatives). In essence, a designer of a recognizer makes a decision between the relative cost of an unintentional occurrence of the gesture and the cost of a user being unable to perform the gesture.

We address the challenge of non-activations by creating a "safe-landing" for users. If a user-performed gesture does not meet a strict threshold, we then consider the gesture using a looser threshold – a more permissive criterion value – and wait to see if a similar motion follows it. The system will recognize a gesture either if the end-user performs a tightly thresholded motion gesture (i.e. success in the first instance), or if the user performs two loosely thresholded gestures within a timeout. This technique is based on our observation of typical user behavior. If a user cannot successfully activate a gesture on the first attempt, their most frequent response is to attempt the gesture again immediately upon recognizing failure. Observing two possible gestures is analogous to observing one highly probable gesture in our model. Essentially, we create a soft-landing for users who attempt a gesture and initially fail. This may reduce user frustration with the system and provide a mechanism for online learning of the gesture set.

In the remainder of this paper we describe our bi-level thresholding technique, including an implementation that uses a hidden markov model approach. Finally, we present the results of a pilot user study evaluating our solution and discuss future work.

Bi-Level Threshold Recognition

We implemented our bi-level threshold technique using a hidden markov model (HMM) approach [1] due to the input's inherently stochastic nature. Though full detail of HMM recognizers is outside the current scope, we provide an overview of our recognizer for replication.

A smartphone senses a motion gesture as a series of time-ordered acceleration (in 3 dimensions) and orientation features (3 degrees of freedom). An HMM is a probabilistic finite state automaton that models the gesture by transitioning from one state to the next. Each state represents a distribution across parameters (acceleration and orientation), and the probabilistic transitions between states represent the likelihood of that state transition. An HMM-based recognizer is comprised of a set of models (one per gesture), each a subset of the states and transitions in the HMM. We use the Viterbi algorithm (see [1] for more detail on the algorithm) to label a candidate gesture with the most likely model that best explains the motion.

Though it scales just as well as other HMM recognizers (e.g. [4]), our recognizer currently accepts three gestures taken from Ruiz et al.'s consensus gesture set [2]: the *Double-flip* gesture, the *Next* gesture (a flick

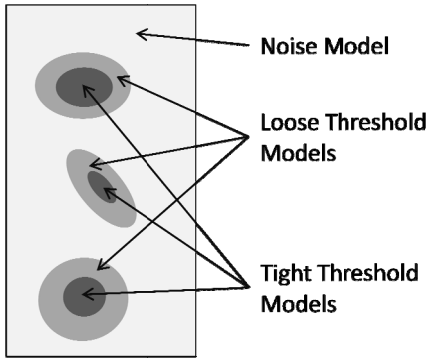

Figure 1. A description of the bi-level thresholding model in the allowable input space of acceleration and orientation.

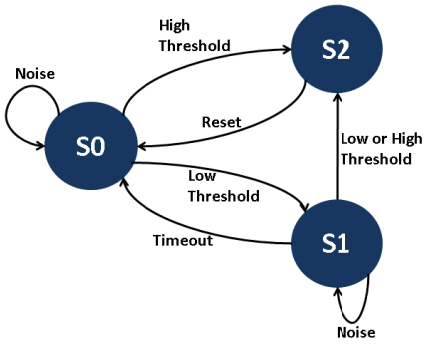

Figure 2. Bi-level thresholding described as a state machine.

of the wrist to the right) and the *Previous* gesture (a flick to the left).

In order to recognize our three gestures, our recognizer includes seven models connected into a single topology in a manner similar to [3]. Three of the models represent accurate, tightly thresholded gestures with low false-positive rates. We create these three models by learning the observation distribution (i.e. the acceleration and orientation feature set) from a small number of pre-segmented template gestures collected from 5 expert users. HMM training is accomplished using the Baum-Welch algorithm [1].

The loose threshold is built by copying the three gesture models learned from our experts and loosening the observation distributions for each state. We do this by applying a linear Gaussian blur to all features in each observation distribution. This produces three additional HMM models that are more permissive, i.e. that allow a greater range of values. We tune the blur to create an acceptable false positive rate for both recognizers. For example, if R is an acceptable false positive rate for the single-threshold models, then $R^{1/2}$ is an acceptable false positive rate for the loose-threshold models. This ensures that the false positive rate for the recognizer is no greater than R.

Finally, we add a seventh model that represents random device motion, or noise. This model was created by repeatedly performing a random walk of the above six models and using the random state transitions to saturate the entire space of allowable inputs with a random motion recognizer.

To guide the reader's intuition of how this recognizer works, consider Figure 1 showing the space of input for acceleration and orientation. The noise model dominates the background region of the Venn Diagram. However, if the observations from the smartphone sensors lie inside the loose threshold, then the loose-threshold models have higher probability. Likewise, if the observations from the sensors lie inside the tight threshold models, then these models dominate.

To recognize a gesture, consider the three-state model in Figure 2. At any point we analyze the inputs from the smartphone. Most frequently, the observations match the noise model, and the recognizer is in state S0. If, however, the observation is recognized by the tight threshold HMM as one of double-flip, next, or previous, we assign that interpretation to the gesture, moving to S2. If, instead, the gesture is accepted by the loose-threshold HMM, we move to state S1. State S1 indicates that one instance of a possible gesture may have occurred. If, within a timeout period, a possible gesture is again observed by our loose-threshold model (or, of course, if a high threshold gesture occurs), we recognize the gesture as its corresponding class, moving to S2. This implementation detects a success if it detects a single good gesture or two consecutive potential gestures. We maintain a high barrier of initial entry but make use of temporal cues to infer user intent and provide a recognition safety net.

Pilot Study

We performed a pilot study gauging the effectiveness of the bi-level thresholding. We asked 8 participants to perform gestures with two Nexus One smartphones: one equipped with our bi-level threshold recognizer and the other with a single tight threshold recognizer (order

in which the phones were given was counterbalanced). Each participant performed each gesture 41 times for a total of 246 instances over the two conditions. The users were asked to perform one gesture at random and the system only moved on to the next instance if it detected that gesture or if 15 seconds had elapsed. The results support our idea of bi-level thresholding as a safety net. 60% of all gestures recognized by our bi-level threshold recognizer were detected using the double loose model. Though there were no significant difference in the time taken to successfully perform a gesture between the conditions, nor in overall failure rates (6% of the gestures were not successfully performed in the given time in either conditions), the large proportion of successes caught using our secondary threshold gives some promise that the technique acts to improve overall efficiency. If the secondary threshold did not exist, users would be forced to perform the gesture at least once more.

Discussion and Future Work

Though we have found some positives effects of the technique with our participants, our pilot study is too small to detect subtler results. We found that any effects in the success rates and time spent per gesture are confounded by learning effects. Our users were less frustrated using the second phone, regardless of condition (regardless of which recognizer was used)

We are interested in performing a long term user study in a realistic setting to detect frustration levels and emergent behavior. Specifically, we hypothesize that the loose threshold helps users converge to the template gesture over time given feedback. We would like to explore how behavior changes over time and the design of recognizers as tools for realtime training. Additionally, we will consider feedback mechanisms to expose bi-level thresholding in order to improve learnability of both recognizer and gesture set.

Lastly, adaptable recognition thresholds may improve overall failure rates. Continuous motion (e.g. walking) may imply that the tight threshold should tighten to limit false positives. Similarly, users failing to trigger gestures may benefit from a lower loose threshold.

Conclusion

We present a technique to lower the number of failures in the recognition of motion gestures by recording success when users perform either a good gesture or two poor gestures consecutively. Our HMM implementation acts as a safety net for users who struggle activating a tightly bound recognizer. Initial results show that bi-level thresholding merits further study as a way to improve the balance between false positives and false negatives. We hope to study the use of bi-level thresholding as a tool for scaffolding long term learning of motion gestures on mobile devices.

References

[1] Rabiner, L.R. A tutorial on hidden Markov models and selected applications in speech recognition. In *Readings in speech recognition* (1990).

[2] Ruiz, J., Lank E., and Yang L. User-Defined Motion Gestures for Mobile Interaction. In *Proc.of CHI 2011.*

[3] Hyeon-Kyu Lee; Kim, J.H. An HMM-based threshold model approach for gesture recognition. In *Pattern Analysis and Machine Intelligence* (1999). pp.961-973.

[4] Schlömer, T.; Poppinga, B.; Henze, N., and Boll, S. Gesture recognition with a Wii controller. In *Proc of Tangible and embedded interaction 2008.*