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

Matei Negulescu
University of Waterloo
Waterloo, ON, Canada
mnegules@uwaterloo.ca

Jaime Ruiz
Colorado State University
Fort Collins, CO, USA
jgruiz@cs.colostate.edu

Edward Lank
University of Waterloo
Waterloo, ON, Canada
lank@cs.uwaterloo.ca

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 often used to classify motion and effectively balances the rates of false positives and false negatives. We present a bi-level threshold recognition technique designed to lower the rate of recognition failures by accepting either a tightly thresholded gesture or two consecutive possible gestures recognized by a relaxed model. Evaluation of the technique demonstrates that the technique can aid in recognition for users who have trouble performing motion gestures. Lastly, we suggest the use of bi-level thresholding to scaffold the learning of gestures.

KEYWORDS
Bi-level thresholding, motion gestures, safety net

ACM CLASSIFICATION KEYWORDS
H.5.2 [Information Interfaces and Presentation]: User Interfaces - Interaction styles.

GENERAL TERMS
Human Factors

INTRODUCTION
Modern sensors such as accelerometers can be leveraged to expand a mobile device's input space by detecting *motion gestures* – gestures that require a user to move and/or translate the entire device in three dimensions. A unique challenge in the design of motion gestures for mobile interaction is the need to develop recognition algorithms that are sufficiently powerful to both recognize a large set of motion gestures and to discriminate intentional gestures from everyday device movement. The focus of this paper is specifically in discriminating intentional motion gestures from everyday motion. Because smartphones are frequently carried in a purse or pocket, the accelerometers and gyroscopes that measure device movement are frequently receiving data. Without careful tuning, unintended commands (i.e. false positives) can be invoked.

MobileHCI'12, September 21–24, 2012, San Francisco, CA, USA.

There are two possible techniques for segmenting a motion gesture from a smartphone's input stream. The first, and most common, approach is to use an explicit delimiter to discriminate a motion gesture from everyday device movement [6]. Researchers have used hardware buttons, on-screen buttons, and specific, easy-to-discriminate motion gestures as delimiters. However, there are several situations where it is undesirable to use a delimiter. For example, consider using a motion gesture repeatedly to step through a set of objects. Performing an explicit delimiter for each motion gesture may frustrate end-users, particularly if they must repeat a large number of motion gestures within a restricted time. Furthermore, even if delimiters support reliable discrimination from an input stream, it is also important to determine how necessary delimiters are to the design of motion-gesture input.

The second technique for discriminating motion gestures from random device movement is to create a threshold, i.e. a criterion value, that best trades-off between false positives (accidental activations) and false negatives (failed attempts to perform a gesture). If the criterion value is too permissive, many false positives will occur. However, if the criterion value is too restrictive, it may become very difficult for the system to reliably identify intentional user gestures. Designers of systems frequently use visualization techniques like receiver operating characteristic curves to identify the best criterion value for a recognizer (e.g. [1]). Despite this, a majority of the motion gesture research uses delimiters, not criterion values, presumably because of the difficulty of selecting a criterion value that appropriately balances false positives and false negatives [6].

In this paper, we address the challenge of non-activations by creating a novel, bi-level thresholding technique for selecting a criterion value that is appropriately restrictive and while not yielding a prohibitively high number of false negatives. Our bi-level thresholding technique works as follows: if a user-performed gesture does not meet a strict threshold, we then consider the gesture using a relaxed threshold – a more permissive criterion value – and wait to see if a similar motion follows it. Figure 1 displays a Venn diagram that represents our input space. The system recognizes 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 relaxed thresholded gestures within a short period of time.

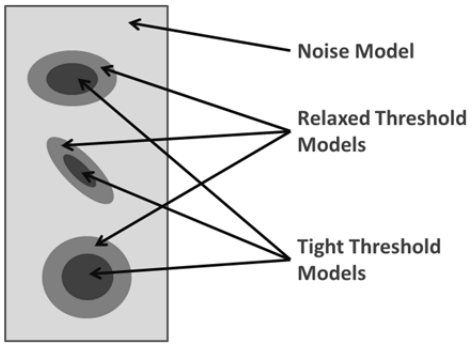

Figure 1. An illustration of the bi-level thresholding model in the allowable input space.

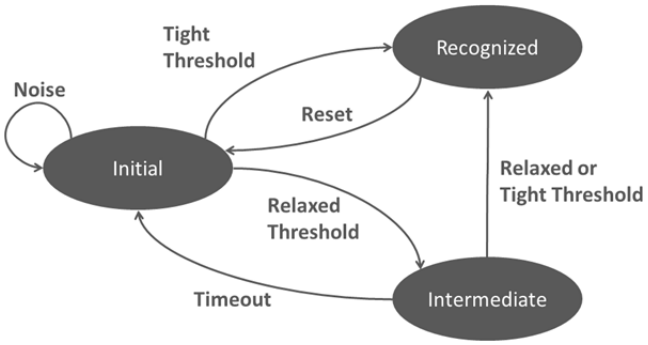

Figure 2. Bi-level thresholding described as a state machine.

RATIONALE FOR BI-LEVEL THRESHOLD RECOGNITION

Our bi-level thresholding technique is based on field notes of typical participant behavior during controlled studies of motion gesture interaction [4]. Participants frequently begin an experiment successfully performing motion gestures. Over time, however, due to the noise within our neurophysiological system, the participant's action may drift from ideal. If a participant does not successfully activate a gesture, their most frequent response is to attempt the gesture again immediately upon recognizing failure. If the gesture succeeds on the second attempt, the cost of missed activation seems quite small. However, participants become frustrated if repeated attempts to activate a gesture also fail. As a result, they begin to alter their input patterns, until they find a gesture profile that is correctly recognized.

We observe that a failure to perform a recognized gesture is often the result of the sensor input falling just outside of the recognition threshold specified by the criterion value. However, adjusting the criterion to recognize these gestures would lead to a prohibitively high rate of false positives.

Our goal in introducing bi-level thresholding was to create a "soft-landing" for users who attempt a gesture and almost succeed at exceeding the criterion value. Rather than requiring the user to repeatedly attempt to match a tight threshold, we noted that the likelihood of observing two sequential gestures at a lower criterion value within a time period is the square of the likelihood of observing one instance of that relaxed-criterion gesture within the time period (e.g., if the odds are 1 in 10 of one relaxed threshold within a time period, the odds of observing two relaxed thresholds within that same time period are 1 in 100). This technique can be seen as a mediation technique that helps to clarify a user's input as outlined by Mankoff et al.[3]. However, in contrast to simple repetition techniques, the goal of bi-level thresholding is adapting to the user's first motion to improve the likelihood of recognition for users who struggle performing motion gestures.

We hypothesized that bi-level thresholding may support successful gesture-from-noise discrimination in situations where a tighter threshold would not. In addition, bi-level thresholding may also provide a mechanism for gradual online learning of the gesture set because users are more likely to succeed at performing correctly recognized gestures.

IMPLEMENTING BI-LEVEL THRESHOLDING

Bi-level thresholding is meant to be recognizer agnostic, applicable to both state based recognizers (e.g. hidden Markov models) and temporal matching algorithms (e.g. Dynamic Time Warping). At a high level, our technique can be visualized as a simple three-state finite state automaton (FSA), as shown in Figure 2. From the *Initial* state, if the recognizer observes a tight-threshold gesture, the system moves to the *Recognized* state and the gesture is recognized. If, in contrast, we observe a relaxed-threshold gesture, the system moves to state the *Intermediate* state. In this state, if the system receives either a tight-threshold or relaxed-threshold input, the system moves to the *Recognized* state and the gesture is recognized. If, instead, a timeout occurs, the system moves back to the *Initial* state.

To evaluate our technique, we implemented bi-level thresholding using a hidden Markov model (HMM) approach [2] due to the input's inherently stochastic nature. The implementation consists of a continuous HMM with four-state models recognizing each gesture. As features, we use time-ordered acceleration (in three dimensions) and orientation readings (three degrees of freedom). Though a full description of hidden Markov models is outside the current scope, we give a brief outline of our gesture models.

The tight-threshold gesture models were built by having six expert users perform each of the gestures in our gesture set 50 times. Though it scales as well as other HMM recognizers, as a proof of concept, our gesture set contains three gestures taken from Ruiz et al.'s consensus gesture set: *Double Flip*, *Next,* and *Previous* (see [5,6]). The HMM was trained using the Baum-Welch algorithm [2] on the pre-segmented gestures performed by our experts. The gesture models comprising our tight threshold were tuned so that three expert users can perform the gestures with greater than 80% accuracy over 20 instances while achieving no more than two false positives when tested with 20 minutes of background walking data.

The relaxed threshold is built by copying the tight-threshold gesture models and loosening the observation distributions for each state 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 the gestures. For example, if $R \in (0, 1)$ is an acceptable false positive rate for the single-threshold models, then $R^{1/2}$ is an acceptable false positive rate for the relaxed-threshold models. This ensures that the false positive rate for all gestures is at most R.

Finally, we added a 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.

As we wish for our recognizer to be always on (as opposed to recognizing pre-delimited content), our implementation links all the gesture and the noise models together with a start state to form a single continuous model, as in [2]. In this method, the Viterbi algorithm is used to consider the most likely sequence of states that generates given sensor observations. A recognized gesture is one where the Viterbi path ends in a gesture model's end state. The interplay of the gesture models and the noise model creates the initial tight threshold that segments noise from gestures.

To guide the reader's intuition of how this recognizer works, consider Figure 1 showing the input space for three gestures. The noise model dominates the background region of the Venn Diagram. However, if the observations from the smartphone sensors lie inside the relaxed threshold, then the relaxed-threshold models have higher probability. Likewise, if the sensor observations lie inside the tight threshold models, then these models dominate. As noted, Figure 2 illustrates how recognition is performed using our relaxed-threshold, tight-threshold, and noise models.

STUDY

We evaluated bi-level thresholding to gauge its feasibility as a safety net for participants who have trouble performing gestures. The study used a within participants design asking eight participants to perform each of three motion gestures, *Double-flip*, *Next*, and *Previous,* 42 times each. A software glitch resulted in the elimination of the first gesture in each set, yielding 125 gestures attempted by eight participants, or 1000 motion gestures in total. A Nexus One smartphone was used to perform and recognize gestures. The order of the gestures was randomized, with the software ensuring that each gesture was performed the same number of times.

The application presented the user with a black screen. In the centre of the screen the word "Double-Flip", "Next", or "Previous" was presented to the user. Once the system detected the given gesture, the screen flashed green, vibrated for 200ms and preceded to the next gesture. If a gesture other than the given gesture was recognized (i.e. a *Recognition Error* occurred), the screen vibrated for 200ms, but did not move on to the next gesture. Instead, it logged the misrecognized attempt and presented the gesture again. If the participant failed to activate the given gesture within 15 seconds, the screen flashed red for 200ms, vibrated, and moved on to the next gesture after logging the instance as a *skipped gesture*.

There are two open questions which we aim to address with our experiment. First, is the bi-level threshold useful in supporting recognition? To analyze this question, we examined at the proportion of gestures recognized using the double relaxed threshold versus the number of gestures recognized using a tight threshold. If you consider the FSA in Figure 2, the tight threshold could successfully recognize gestures from either the *Initial* or *Intermediate* states. In either of these cases, the bi-level threshold provides limited benefit as it does not eliminate gesture attempts. It is unclear how common the bi-level technique will be during recognition. It may be the case that it increases recognition reliability significantly (i.e. it frequently performs the recognition of gestures). On the other hand, participants may hit the tight threshold or fail to hit a double relaxed threshold often enough to improve recognition rates.

Second, we wish to determine whether the rate of skipped gestures can tell us anything about the potential of bi-level thresholding as a mechanism for scaffolding. For example, do users who struggle with motion gestures make increased use of bi-level thresholding?

RESULTS

During our experiment, 93% of gestures were recognized successfully for all users within the timeout period. The other 7% were skipped gestures. Moreover, the recognizer performed at an accuracy of 95.3% (S.D. = 6.4%) with five out of eight participants achieving 99.8% accuracy. Due to the tuned tight threshold and the given task, no false positives were recorded during the session.

We analyze the behavior of our recognizer by examining the various recognition paths taken through the FSA (Figure 2). For each user, we separate the proportion of successes recognized with a) the double relaxed threshold, b) the single tight threshold, and c) those recognized with the relaxed then tightly thresholded models (i.e. tight from the *Intermediate*). We show the various fractions in Figure 3.

We note that 65% of recognized gestures were recognized using the double relaxed gesture model. Another 9% of successful recognitions were flagged by the single tight threshold model while already in the *Intermediate* state, i.e. after an initial relaxed threshold event. The remaining 26% of gestures were recognized using the single tight threshold, from the *Initial* state. The large proportion of successes caught using our bi- threshold gives some promise that the technique acts to improve overall efficiency.

While an average rate of 7% of total gestures attempted using the bi-level threshold recognizer (SD=10.0) were

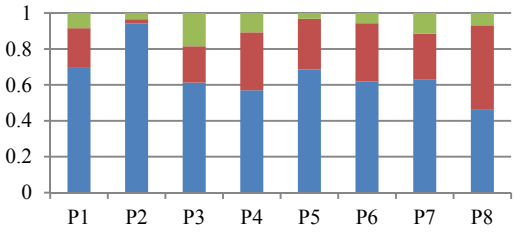

Figure 3. Breakdown of gestures by whether recognition occurred using the double relaxed model (blue), the single tight model (red), or the relaxed then tight model (green).

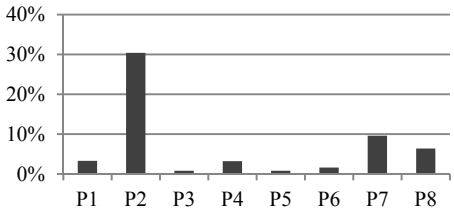

Figure 4. Rate of skipped gestures, per participant

skipped gestures, there is variability among participants. As shown in Figure 4, P2 is an outlier, failing to successfully perform the gesture within the timeout window 30% of the time. P2 provides an interesting case study for our bi-level thresholding technique. P2 only successfully completed 70% of the gestures, compared to over 90% for all other participants. Of P2's successfully recognized gestures, 94% were recognized using our double-relaxed model (Figure 3) versus 50-66% for all other participants. P2 was an outlier in both the failure rate experienced performing motion gestures and in how frequently their successes were recognized using the bi-level thresholding technique. Although care must be taken to generalize from the behavior of this one participant, this participant's case study may indicate the potential use of the lower threshold for users who have trouble completing motion gestures.

DISCUSSION

The results of our experiment indicate that the bi-level thresholding technique for motion gesture recognition can aid in recognition of motion gestures. The recognizer was tuned so that expert users found it easy to use while limiting the number of false positives. However, though recognition rates were high, our novice participants had difficulty in consistently activating the motion gestures. Two-thirds of all gestures recognized during our experiment were recognized as a result of the double-relaxed threshold model of the bi-level thresholding technique. These gestures would need at least another attempt by the participant to be recognized successfully if using a similarly tuned single-tight threshold. The relaxed threshold acted as a safety net for participants who were inexperienced with motion gestures. An open question is whether such a safety net can be used to scaffold learning to perform motion gestures over a long period of time.

We note that, in our experiment, we do not consider delimiters. It is true that, if an effective delimiter is chosen, then the system can use a more permissive criterion function -- one less likely to label movement as noise. However, even with delimiters, criterion functions are still required. To understand why, consider what happens when a delimiter is activated. If, after delimiter activation, the end-user is performing any action at all – walking, driving, looking at the phone, turning in a desk chair, holding the phone, etc. – then the accelerometer and gyroscope in the smartphone will receive data. If the system assumes that all received data constitutes a motion gesture, then a gesture may fire accidentally before the user has any chance to execute a deliberate gesture, solely based on random input. Regardless of whether or not delimiters are used, it is still necessary to create a noise model and to select a criterion value to separate deliberate movement from noise.

Finally, while we focus on motion gestures as the application space for bi-level thresholding, we believe that the technique is applicable to other gestural domains, including surface gestures and stylus interfaces. For example, consider using an always available scratch-out gesture within a stylus program like Windows Journal. If the scratch-out gesture fails initially (something that frequently occurs), a bi-level recognition model could be designed that handles two poorly performed scratch-out gestures, thus aiding reliable recognition.

CONCLUSION

We present a bi-level thresholding technique to support the recognition of motion gestures for smartphone input. Our results show that, when available, the bi-level thresholding technique frequently catches input gestures that an optimized criterion function misses. The end result is that end-users need fewer attempts to successfully activate motion gestures using bi-level thresholding.

ACKNOWLEDGEMENTS

Funding provided by the Natural Science and Engineering Research Council of Canada (NSERC) and the Networks of Centres of Excellence for Graphics, Animation and New Media (NCE-GRAND).

REFERENCES

1. Fogarty, J., Baker, R.S., and Hudson, S.E. Case studies in the use of ROC curve analysis for sensor-based estimates in human computer interaction. *Proc. of GI*, (2005), 129–136.
2. Lee, H.K. and Kim, J.H. An HMM-based threshold model approach for gesture recognition. *IEEER Trans. on Pattern Analysis and Machine Intelligence,* 10 (1999), 961–973.
3. Mankoff, J., Hudson, S.E., and Abowd, G.D. Interaction techniques for ambiguity resolution in recognition-based interfaces. *Proc. of UIST '00*, ACM (2000), 11–20.
4. Negulescu, M., Ruiz, J., Li, Y., and Lank, E. Tap, swipe, or move: attentional demands for distracted smartphone input. *Proc. of the AVI*, ACM (2012), 173–180.
5. Ruiz, J., Li, Y., and Lank, E. User-defined motion gestures for mobile interaction. *Proc. of CHI*, ACM (2011), 197–206.
6. Ruiz, J. and Li, Y. DoubleFlip: a motion gesture delimiter for mobile interaction. *Proc. of CHI.* ACM (2011), 2717–2720.