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

Matei Negulescu
University of Waterloo
mnegules@cs.uwaterloo.ca

Jaime Ruiz
University of Waterloo
jgruiz@cs.uwaterloo.ca

Edward Lank
University of Waterloo
lank@cs.uwaterloo.ca

Technical Report CS-2011-25

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 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

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 the entire smartphone device in three dimensions. A unique challenge in mobile interaction design around motion gestures is the need to develop recognition algorithms that are sufficiently powerful to both recognize a large set of motion gestures and to discriminate intentional motion 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, they move with the user, the accelerometers and gyroscopes that measure device movement and acceleration are frequently receiving data. Without careful tuning, unintended commands (i.e. false positives) can be invoked.

There are two possible techniques for segmenting a motion gesture from a smartphone's input stream. The first, and most common, approach is to make use of an explicit delimiter to discriminate a motion gesture from everyday device movement [3]. Researchers have used hardware buttons, on-screen buttons, and specific, easy-to-discriminate motion gestures as delimiters. However, there are many situations where it may be undesirable to use a delimiter. For example, consider uses a motion gesture repeatedly to step through a set of objects. Performing an explicit delimiter for each motion gesture may annoy 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 interesting to determine exactly how necessary delimiters are to the design of usable 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 from its input stream. The designers of systems frequently use visualization techniques like receiver operating characteristic (roc) curves to identify the best criterion value for a recognizer (e.g. [1]). Despite this, almost all current research into motion gestures uses delimiters, not criterion values, presumably because of the difficulty of selecting a criterion value that appropriately balances false positives and false negatives [3]

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 both appropriately restrictive and yet does not result in 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 looser 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

CHI 2012, May 5-10, 2012, Austin, TX, USA.

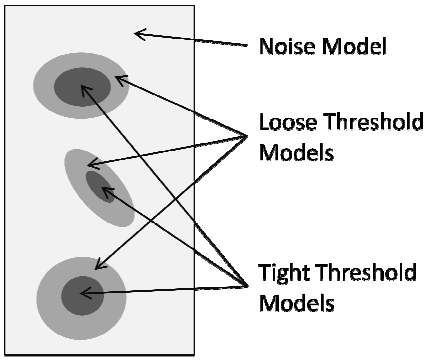

Figure 1. An illustration of the bi-level thresholding model in the allowable input space.

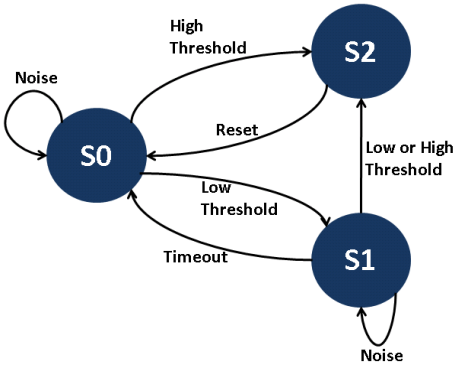

Figure 2. Bi-level thresholding described as a state machine.

input space. 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.

RATIONALE FORBI-LEVEL THRESHOLD RECOGNITION

Our bi-level thresholding technique is based on field notes of typical participant behaviour during controlled studies of motion gesture interaction. 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 on the first attempt, his or her 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, if repeated attempts to activate a gesture or button fail, the participants become frustrated. They begin to alter their input patterns, seeking to change the kinematics of their input until they find a gesture profile that succeeds. They frequently land in states where failures are highly common until they diagnose why they are failing to activate the gesture and correct.

Many times we find that a failure to activate is a result of sensor input falling just outside of the threshold for recognition specified by the criterion value. Recognizing these motion gestures would boost our false positive rate to prohibitively high levels.

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 loose-criterion gesture within the time period (e.g. if the odds are 1 in 10 of one loose threshold within a time period, the odds of observing two loose thresholds within that same time period are 1 in 100).

We hypothesized that bi-level thresholding may support successful gesture-from-noise discrimination in situations where a tighter threshold would not. Because users are more likely to succeed at attempting gestures, bi-level thresholding may also provide a mechanism for gradual online learning of the gesture set. Users will be more likely to accomplish their gesture within two attempts with the bi-level threshold enabled than without, so they will continue to experiment with motion gestures.

IMPLEMENTING BI-LEVEL THRESHOLDING

Conceptually, our recognition algorithm can be visualized as a simple three-state finite state automaton, as shown in Figure 2. From an initial state, S0, if the recognizer observes a high-threshold gesture, the system moves to S2 and the gesture is recognized. If, in contrast, we observe a lower-threshold gesture, the system moves to state S1. In this state, if the system receives either a high-threshold or low-threshold input, the system moves to state S2 and the gesture fires. If, instead, a timeout occurs, the system moves back to state S0.

We implemented our bi-level threshold technique using a hidden markov model (HMM) approach [2] 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 [2] for more detail on the algorithm) to label a candidate gesture with the most likely model that best explains the motion.

We create tight-threshold gesture models by having six expert users perform each of the three 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 [4]). Expert users pre-segment the gestures from input noise using an on-screen button. The expert user presses the button and then performs the gesture. We train the HMM on the acceleration and orientation feature set of these pre-segmented gestures using the Baum-Welch algorithm [2].

The loose threshold is built by copying the tight-threshold 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 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 loose-threshold models. This ensures that the false positive rate for all gestures is at most R.

Finally, we add 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.

To guide the reader's intuition of how this recognizer works, consider Figure 1 showing the space of input for acceleration and orientation 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 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. As noted above, Figure 2 shows how recognition using our low-threshold, high-threshold, and noise signals is performed for each individual gesture.

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 8 participants to perform each of three motion gestures, *Double-flip*, *Right Flick*, and *Left Flick* 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 Double-Flip, Right Flick and Left Flick were random, with the software simply ensuring that each gesture was performed the same number of times.

The application presents the user with a black screen. In the centre of the screen the word "Double-Flip", "Next", or "Previous" is presented to the user, where "Next" corresponds to Flick Right and "Previous" corresponds to Flick Left. Once the system detects the given gesture, the screen will flash green, vibrate for 200ms and move on to the next gesture. If a gesture other than the given gesture is recognized (i.e. a *Recognition Error* occurs), the screen vibrates for 200ms, but does not move on to the next gesture. Instead, it logs the misrecognized attempt and presents the gesture again. If the participant fails to activate the appropriate gesture within 15 seconds, the screen will flash red for 200ms, vibrate, and move 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 look at the proportion of gestures recognized using the double loose 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 of states S0 or S1. 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, it may also be the case that the bi-level threshold is never activated. Participants may hit the tight threshold or fail to hit a double loose threshold so frequently that the bi-level technique is insignificant in improving recognition.

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

RESULTS

During our experiment, 93% of gestures were recognized successfully for all users within the timeout period. The other 7% were skipped gestures, i.e. gestures which were not recognized before the 15 second timeout.

We analyze the behaviour of our recognizer by analyzing the various recognition paths taken through the FSA pictured in Figure 2. When considering correctly recognized gestures, we separate, for each user, the proportion of recognized gestures that were recognized with the double loose threshold versus the single tight threshold versus the loose then tight threshold (i.e. tight from S1). We show the various fractions in Figure 3.

We note, first, that 65% of recognized gestures were recognized using the double loose gesture model. Another 9% of successful recognitions were flagged by the single tight threshold model while already in state S1, i.e. after an initial loose threshold event. The remaining 26% of gestures were recognized using the single tight thresholded, from state S0. The large proportion of successes caught using our double loose 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 skipped gestures, there is some variability among our participants. In particular, Figure 4 shows that 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, as compared to over 90% for all other participants. Of P2's successfully recognized gestures, 94% were recognized using our double-loose model (Figure 3), versus, typically, between half and two-thirds for other participants. P2 was an outlier in both the failure rate experienced performing motion gestures and in how frequently the gestures that were recognized were recognized with the bi-level thresholding technique. Though care must be taken to generalize from the behaviour of this one participant, this participants' case study may indicate the potential use of the lower threshold for users who have trouble completing motion gestures.

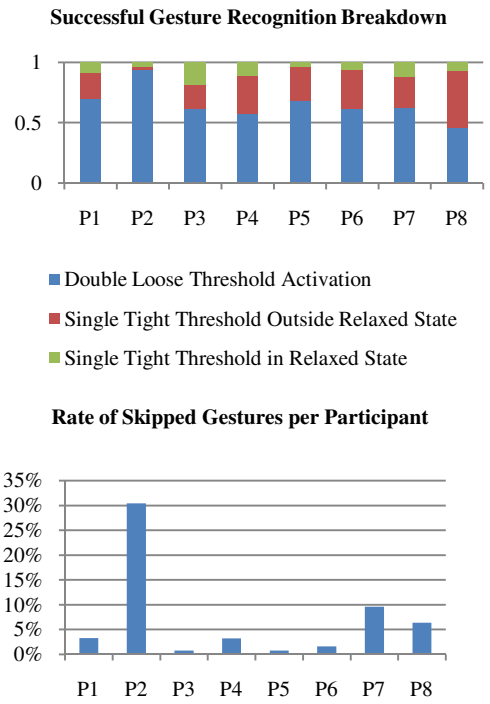

Figure 3.(Top) Breakdown of gestures successfully recognized by whether recognition occurred using the double loose model (blue), the single tight model (red) or the loose-then-tight model. (Btm) Rate of skipped gestures, per participant

DISCUSSION

The results of our experiment indicate that the bi-level thresholding technique for motion gesture recognition can aid in recognition of motion gestures. Two-thirds of all gestures recognized during our experiment were recognized as a result of the double-loose threshold recognition 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 single-tight threshold recognizer.

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 this is the case, 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 – 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 their 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 thresholds, we believe that they are applicable to other gestural domains, including surface gestures and stylus interfaces. As a simple example of another domain where a bi-level threshold might aid recognition, consider using a 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 of always-available gestures like scratch-out.

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.

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. In *Proc of Graphics Interface 2005.* 129–136.
2. Rabiner, L.R. A tutorial on hidden Markov models and selected applications inspeech recognition. *Proceedings of the IEEE 77*, 2 (1989), 257-286.
3. Ruiz, J. and Li, Y. DoubleFlip: a motion gesture delimiter for mobile interaction. In *Pro cof CHI 2011,* 2717–2720.
4. Ruiz, J., Li, Y., and Lank, E. User-defined motion gestures for mobile interaction. In *Proc.CHI 2011*, 197–206.