Probabilistic Palm Rejection Using Spatiotemporal Touch Features and Iterative Classification

Julia Schwarz Robert Xiao Jennifer Mankoff Scott E. Hudson Chris Harrison
Human-Computer Interaction Institute
Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213
{julia.schwarz,brx,jmankoff,scott.hudson,chris.harrison}@cs.cmu.edu

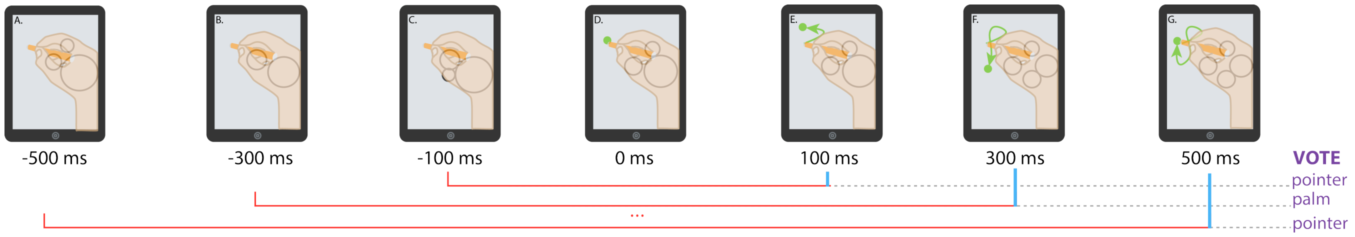

Figure 1. An illustrated example of touches present at different points in time relative to a touch contact of interest (D, green dot). Touch points due to palms (hollow circles) are often ephemeral, large, and have low velocity. Our approach extracts features and performs classification of each touch point at several points in time (blue lines), using different sized time windows (red). In this example, we show how the classification for the green dot only changes (purple text) as the window size changes.

ABSTRACT
Tablet computers are often called upon to emulate classical pen-and-paper input. However, touchscreens typically lack the means to distinguish between legitimate stylus and finger touches and touches with the palm or other parts of the hand. This forces users to rest their palms elsewhere or hover above the screen, resulting in ergonomic and usability problems. We present a probabilistic touch filtering approach that uses the temporal evolution of touch contacts to reject palms. Our system improves upon previous approaches, reducing accidental palm inputs to 0.016 per pen stroke, while correctly passing 98% of stylus inputs.

Author Keywords: Palm rejection; touchscreen; tablet computing; touch interaction; pen and stylus input.

ACM Classification Keywords: H.5.2 [Information interfaces and presentation]: User Interfaces

INTRODUCTION
Tablet computers are often called upon to emulate classical pen-and-paper input. However, most touch devices today lack palm rejection features – most notably the highly popular Apple iPad tablets. Failure to reject palms effectively in a pen or touch input system results in ergonomic issues [3], accidental activation and unwanted inputs, precluding fluid and efficient use of these input systems. This issue has been well explored in the academic literature (see e.g., [10,11,12,22]).

The contributions of this work are three-fold. Foremost, we describe a novel, probabilistic approach to palm rejection. Our system requires no initial configuration and is independent of screen orientation and user handedness. Second, we review contemporary palm rejection implementations and compare our approach against two applications in a user study, offering the first publicly available comparison of such systems. Through our user study, we show that our implementation offers equal or superior performance to these applications.

We prototyped our approach on an Apple iPad 2 running iOS 6 – a platform without native palm rejection or stylus input. Our approach, however, is platform agnostic and will work on any system that reports multiple touch contacts along with location and touch area.

SPATIOTEMPORAL TOUCH FEATURES
Our work began with a series of observations of stylus use on tablets. We identified five properties that distinguished palms from pointer (i.e., finger or stylus) inputs: 1) the touch area for palms tends to be large, whereas pointers have small tips; 2) on most touchscreens, the large palm contact area is segmented into a collection of touch points, which often flicker in and out; 3) these palm points tend to be clustered together, whereas the pointer is typically isolated; 4) stylus touches have a consistent area, unlike palms, which change in area as they deform against the screen; and 5) palms generally move little, while pointer inputs tend to have longer, smoother trajectories.

Another insight was that there was often significant context that existed before a touch point appeared on the screen. For example, when dotting an ‘i’ the stylus touch might only exist for 50ms – however, the palm might have been on the display for several seconds beforehand. As our approach records all touch data, we can look backwards in time to make a more confident classification.

Using our observations as a starting point, we derived a series of features that characterize touch points of interest and their relationships to neighboring points[1]. These features are computed over touch event sequences corresponding to a particular touch contact (which we will eventually need to categorize as either a stylus or part of a palm) and occurring over windows of time, centered at *t*=0 (i.e. birth of the touch point). We expand the time window symmetrically about *t*=0, ensuring that data from before and after the initial touch event are included (Figure 1).

Each touch event has a centroid position and a radius (indicating the maximum distance from the centroid to the perimeter of the touch area). Our features consist of statistics (mean/stdev/min/max) computed over sequences of touch events corresponding to a particular touch contact for each time window. We calculate these statistics for the radius of each event and speed and acceleration of consecutive events. Additional features include the total number of events in the sequence and mean/stdev/min/max calculated over the Cartesian distances between the centroid of the touch event at *t*=0 and all touch events in any concurrent sequences (belonging to other touch contacts). All of these features are rotation and flip invariant. This should minimize the effect of device and hand orientation, as well as handedness, on classification.

Similar features have been used for other applications, including finger angle estimation [25], thumb-driven interactions [1] and more generally, finger pose estimation [20]. Wang and Ren provide a more complete overview of possible finger properties and related work [24].

To better understand which features discriminate palm from stylus, we performed feature selection on our training dataset (11,373 instances collected from 3 people) using correlation-based feature subset selection [8] with best first search, provided in Weka [9]. We found that min distance to other touches, number of touch events, and min/mean/max/stdev of touch radius to be most valuable[1].

ITERATIVE CLASSIFICATION AND VOTING

Our algorithm records all touch events reported by the touchscreen. After a touch point has been alive for at least 25ms, the system classifies the touch as either "pointer" or "palm". If a touch terminates before 25ms has elapsed, it is classified using all available data. At 50ms after birth, another classification is performed. For every 50ms thereafter, up to 500ms since birth, this classification repeats – each time contributing a single "vote". A temporary touch type, either pen or palm, is assigned based on the majority of the votes accumulated thus far. After 500ms, or if the touch point disappears (whichever comes first), voting stops, and the final vote is used to assign a permanent classification. Note that the vote implicitly encodes a confidence score that can be used in probabilistic input systems (such as those described in [21]).

[1] See `calc_features.cpp` and `classify.cpp` in the supplementary material for information about features and implementation details.

One benefit of our iterative classification approach is that it allows our system to show immediate feedback to the user. The system initially shows its best guess (roughly 98% accurate, see Figure 2) and refines this later as more information becomes available. For example, if a touch is initially guessed to be a pen, the application will render a stroke on canvas. If this guess is later changed, the stroke is removed from the canvas.

TRAINING THE CLASSIFIERS

We trained eleven decision trees using the features described in the previous sections with window sizes ranging from 50 to 1000ms (i.e. classifiers triggered at 25ms, 50ms, 100ms, 150ms, etc. up to 500ms; see Figure 1). Each tree was trained using touch features from all window sizes up to the maximum window size. For example, the classifier triggered at 200ms uses features obtained from window sizes of 50, 100, 200, 300 and 400ms (windows are symmetric, centered on *t*=0). We used Weka [9] to train our decision trees using the C4.5 algorithm [19].

We collected training data using a custom iOS application. For each training instance, a 1cm radius dot was randomly placed on the screen. Users were told to place their palms on the screen however they saw fit, such that they could draw a stroke of their choosing starting in this circle. This procedure allowed us to collect labeled pointer and palm point data. In total, we captured 22,251 touch event instances (of which 2143 are stylus strokes) from five people using a variety of hand poses, tablet orientations, and handedness.

To estimate the effectiveness of our iterative approach, we split our data into 11,373 training instances (from 3 people) and 10,878 test instances (from 2 others). Figure 2 shows test accuracy over increasing time windows. Classification at *t*=1ms is included to approximate instantaneous classification. Accuracy improves as window size increases, plateauing around 99.5% at 200ms. We continued classification out to 500ms for experimental reasons, but as Figure 2 shows, the main gains occur in the first 100ms. This result underscores the importance of leveraging temporal features and also delaying *final* classification.

As shown in Figure 2, performing classification instantly (at *t*=1ms) yields a classification accuracy of 98.4% (kappa=0.79). This is sufficiently accurate that real-time graph-

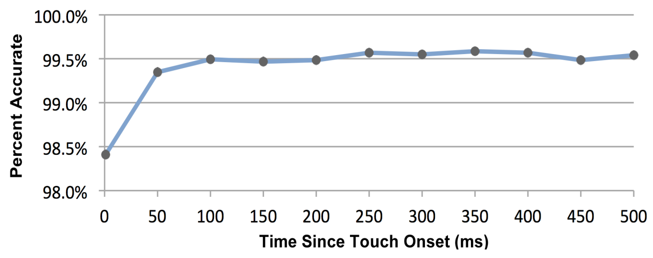

Figure 2. Classification accuracy (true positives) over different durations of time. Leftmost point is at t=1ms.

ical feedback can be rendered immediately while only occasionally requiring later reversion. Reclassifying at 50ms reduces errors by 44%. By continuing iterative classification and voting up to 100ms, accuracy increases to ~99.5% (kappa=0.94), cutting the error rate by a further 29%.

RELATED SYSTEMS

Many palm rejection approaches – utilizing hardware, software, and combinations of the two – have been created, which we now review to position our work.

Hardware Approaches

The most reliable way to disambiguate stylus input from human input is to use special hardware. For example, ultrasonic transducers can be placed at the periphery of a screen to sense ultrasonic pulses emitted by an active pen (see e.g., [16]). It is also possible to use an infrared emitting pen and two or more cameras to triangulate the planar position on a screen (see e.g., iPen 2 [13]). The Jot Touch [14] uses a passive capacitive tip, which simulates a finger touch. The pen itself is powered and pressure sensitive, sending data to the device over Bluetooth. With timing information, it is possible to associate touch events with pen down events.

Another approach, popularized by Wacom, uses resonance inductive coupling [5], which uses a special pen and sensor board that operates behind the conventional capacitive touchscreen. This technology is used in devices such as the Microsoft Surface and Samsung Galaxy Note. Similarly, Gauss-Sense [15] uses a grid of Hall effect sensors behind the touchscreen to sense the magnetic tip of a special pen. LongPad [7] used a grid of infrared proximity sensors and computer vision to separate palm and finger inputs. Finally, advanced capacitive touchscreens can differentiate passive styli by looking at contact size and capacitive properties [2].

Even with special hardware for stylus support, simply distinguishing pen from finger is insufficient if the finger can still be used for input. In this case, unwanted palm touches may still be interpreted as finger touches in the absence of the pen. Thus, software is still needed to reliably distinguish pens and fingers from palms, which the above solutions do not address.

Software Approaches

Although special styli tend to offer excellent precision, a significant downside is the need for a special purpose accessory, which is often platform-specific. Further, additional internal hardware is often required to support these pens, adding to the build cost, size and power draw of mobile devices. Thus, a software-only solution, which can be easily deployed and updated, is attractive. Further, software solutions offer the ability to disambiguate between finger and palm input. However, without an innate way to disambiguate touch events, software solutions must rely on clever processing or interaction techniques.

For optical multi-touch devices, one approach is to identify palm regions visible from the camera image [6]. On mobile devices with capacitive screens, the task is more challenging, since applications generally do not have access to a hand image, or even the capacitive response of the touch screen. Instead, applications must rely on information about touch position, orientation (if available), and size. There are dozens of applications in the iOS and Android app stores that claim to have palm rejection features. Unfortunately, implementations are proprietary, precluding direct analysis.

One method applications employ is to specify a special 'palm rejection region' where all touches are ignored [17], though this is unwieldy. Unfortunately, palm touches outside the input region can still provide accidental input (e.g. accidental button presses). Vogel *et al.* [23] makes use of a more sophisticated geometric model to specify the rejection region, providing a five-parameter scalable circle and pivoting rectangle, which captures the area covered by the palm better than a rectangular region.

A second approach uses spatiotemporal features – looking at the evolution of touch properties and movement over a short time window. We hypothesize that applications that first draw, then remove strokes, must wait some period of time before detecting accidental touches. Two applications exhibiting this behavior include Penultimate [18] and Bamboo Paper [2]. Both applications require the user to specify information their handedness and use the tablet in a fixed orientation, neither of which our method requires. Additionally, Penultimate requires users to specify one of three handwriting poses they use.

USER STUDY

To assess the performance of our palm rejection approach, we compared against Penultimate and Bamboo Paper. As of September 2013, both of these apps have been featured in the Apple App Store, and were subjectively judged by the authors to have the best palm rejection out of 10 candidate applications tested.

We recruited 10 participants from our lab (3 female, one left-handed, mean age 29), who were paid $5 for their time. Users were provided a passive, rubber-tipped stylus, which is the most popular style for use with the iPad. The task was to replicate 15 symbols presented on cards. Participants were instructed to draw each symbol with a single stroke. If the application missed the stroke, they were told to continue to the next symbol. They were allowed to rest their hands on the screen, and to lift, slide and otherwise reposition their palm however they saw fit during drawing.

Six symbol sets, representing a variety of 1D and 2D shapes (the letter 'S', a circle, a dot, a horizontal and vertical line, and the letter 'L'), were presented in random order. This procedure was repeated for the three applications – Bamboo, Penultimate, and our own – in a random order. Bamboo and Penultimate were each configured for the user's handedness and preferred handwriting pose before the experiment; our application did not require configuration.

After each symbol set was drawn, the experimenter recorded the number of strokes that were successfully drawn (true positives), as well as the number of extraneous strokes (typically under the palm) that were drawn (false positives).

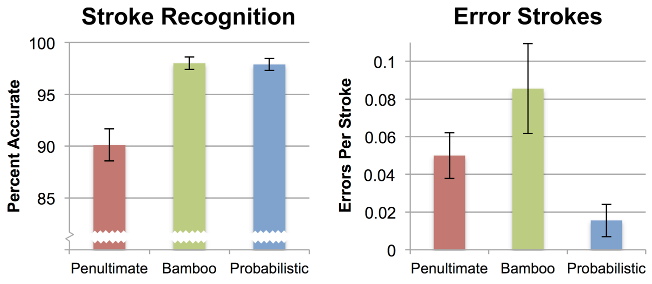

Figure 3. Stroke recognition accuracy and errors per stroke results from our study. Error bars reflect standard error.

This procedure provided 90 stroke attempts per user per application, for a total of 2,700 strokes. We did not record true/false positives for palm classification because it was not feasible to collect ground truth palm data.

RESULTS

Our approach has a true positive rate of 97.9%, compared to Bamboo's 98.0% and Penultimate's 90.1% (Figure 3). This outcome is not significantly different from Bamboo Paper, but both Bamboo and our approach are significantly more accurate than Penultimate ($p < 0.05$). Statistical significance was assessed by running a repeated measures ANOVA ($F_{2,18} = 20.53$, $p < 0.05$), followed by a Tukey HSD test.

Additionally, our approach has fewer false positives than Bamboo and Penultimate: 0.016 errors/stroke vs. 0.086 and 0.050 respectively (Figure 3). The difference between our approach and Penultimate was not significant, though our false positive rate was significantly lower than Bamboo (Tukey HSD $p < 0.05$; ANOVA $F_{2,18} = 5.09$, $p < 0.05$).

Although our system performs with accuracy equivalent to Penultimate, it does not require information about hand position (unlike Penultimate). We believe two things contributed to this robustness: we collected training data that represented a wide range of poses; and, as mentioned above, we designed our feature set to be hand invariant.

CONCLUSION

In this work, we described a palm rejection technique utilizing temporal features, iterative classification, and probabilistic voting. We demonstrate the efficacy of our solution with an evaluation, which showed improvements over popular applications considered to be the current state of the art. Finally, our approach provides a basis for future research efforts in palm rejection.

ACKNOWLEDGEMENTS

This work was funded by NSF Grant IIS1217929, and fellowships from Microsoft Research, Google and Qualcomm.

REFERENCES

1. Boring, S., Ledo, D., Chen, X., Marquadt, N., Tang, A. and Greenberg, S. The fat thumb: using the thumb's contact size for single-handed mobile interaction. In *Proc. MobileHCI '12*, 39-48.
2. Bamboo Paper. Wacom. http://bamboopaper.wacom.com.
3. Camilleri, M., Malige, A., Fujimoto, J., Rempei, D. (2013). *Touch Displays: the effects of palm rejection technology on productivity, comfort, biomechanics, and positioning.* In Ergonomics. Taylor & Francis Group.
4. ClearPad^{TM} Series 3. http://synaptics.com/solutions/products/clearpad
5. EMR® Technology. Wacom. http://www.wacom-components.com/english/technology/emr.html.
6. Ewerling, P., Kulik, A, Froehlich, B. Finger and hand detection for multi-touch interfaces based on maximally stable extremal regions. In *Proc. ITS '12*, 173-182.
7. Gu, J., Heo, S., Han, J., Kim, S. and Lee, G. LongPad: a touchpad using the entire area below the keyboard of a laptop computer. In *Proc. CHI '13*, 1421-1430.
8. Hall, M.A. Correlation-based Feature Subset Selection for Machine Learning. Ph.D. Thesis, 1998. Hamilton, New Zealand.
9. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1), 10-18.
10. Hinckley, K. and Sinclair, M. Touch-Sensing Input Devices. In *Proc. CHI '99*, 223-230.
11. Hinckley, K., Wigdor, D., (2012). Input Technologies and Techniques (Chapter 9). In The Human-Computer Interaction Handbook, 3rd Edition, published by Taylor & Francis.
12. Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H., and Buxton, B. Pen + touch = new tools. In *Proc. UIST '10*, 27-36.
13. iPen 2. Cregle Inc. http://www.cregle.com/pages/pressure-sensitive-stylus-for-your-imac-and-ipad.
14. Jot Touch. Adonit . http://adonit.net/jot/touch
15. Liang, R., Cheng, K., Su, C., Weng, C., Chen, B. and Yang, D. GaussSense: attachable stylus sensing using magnetic sensor grid. In *Proc. UIST '12*, 319-326.
16. MyNote Pen. http://mynote.eu/mynotepen-en.html.
17. Notability Ginger Labs. http://www.gingerlabs.com.
18. Penultimate. Evernote. http://evernote.com/penultimate.
19. Quinlan, J. R. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, 1993.
20. Rogers, S., Williamson, J., Stewart, C. and Murray-Smith, R. AnglePose: robust, precise capacitive touch tracking via 3D orientation estimation. In *Proc. CHI '12*, 2575-2584.
21. Schwarz J., Hudson S., Mankoff, J. and Wilson, A.D. A framework for robust and flexible handling of inputs with uncertainty. In *Proc. UIST '10*, 47-56.
22. Steimle, J. (2012). Survey of Pen-and-Paper Computing. In *Pen-and-Paper User Interfaces* (pp. 19-65). Springer Berlin Heidelberg.
23. Vogel, D., Cudmore, M., Casiez, G., Balakrishnan, R. and Keliher, L. Hand occlusion with tablet-sized direct pen input. In *Proc. CHI '09*, 557-566.
24. Wang, F. and Ren, F. Empirical evaluation for finger input properties in multi-touch interaction. In *Proc. CHI '10*, 1063-1072.
25. Wang, F., Cao, X., Ren, X. and Irani, P. Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In *Proc. UIST '09*, 23-32.