Implicit Palm Rejection using Real-time Hand Model Filters on Tablet Devices

Riyeth P. Tanyag
Ubiquitous Computing Laboratory
Electrical and Electronics Engineering Institute
University of the Philippines-Diliman
rptanyag@upd.edu.ph

Rowel O. Atienza
Ubiquitous Computing Laboratory
Electrical and Electronics Engineering Institute
University of the Philippines-Diliman
rowel@eee.upd.edu.ph

***Abstract*—Most tablet devices usually suffer from the "palm rejection problem", where unintended multiple touches while writing often cause erroneous application behavior and unsightly imprints on the display. We designed a real-time implicit palm rejection algorithm based on hand model filters and touch characteristics. We focus our approach on accurately determining the context of the touch in real-time to provide users with an experience that is close to natural handwriting as possible. Our algorithm uses a model-based filtering method to define the palm rejection region and automatically adjust to palm-first or write-first scenarios. Our implementation can correctly filter 99% of stylus touches with a low errant touch rate of 0.87%. In summary, our palm rejection algorithm is shown to be comparable or better than most currently available note-taking applications.**

Keywords—handwriting; tablet device; palm and arm rejection; errant touches; writing; algorithm; pen-based application; notetaking

I. INTRODUCTION

Tablet devices have found a unique use as a digital alternative to traditional pen and paper for writing, drawing and note-taking. Apart from employing a smooth writing algorithm for note-taking applications, a note-taking application must also have palm rejection capability -- it must be able to reject unintended touches from the user's palm and arm. Most tablets, even the popular *Apple iPad*, do not have built-in palm rejection capability. Without effective palm rejection, unintended touches imprint unsightly marks on the display or trigger erroneous behavior on most handwriting applications. Users are then forced to hover their hands over the tablet, or orient their hands in unnatural and awkward positions, affecting the quality of user experience [1,2,3]. For users, it is important to maintain the feeling of writing naturally on the device in order to fully embrace a tablet device as an alternative to the pen and paper. Most implementations that address the palm rejection problem do not provide a natural and unobtrusive method of rejecting unintended touches, requiring users to use third-party hardware like active digitizer pens or adjust widgets to define writing areas for software applications.

The main objective of our research is to develop a real-time implicit palm rejection algorithm that uses a series of hand model filters to distinguish the writing touch from the unintended touches. Unlike some palm rejection algorithms that implement stroke recovery to improve accuracy, we focus our approach on accurately determining the context of the touch in real-time. This is to provide users with an experience that is close to natural handwriting as possible. To determine the context of a touch, decisions were made based on touch characteristics and hand model filters to address write-first and palm-first scenarios. Unlike explicit palm rejection techniques, our algorithm does not require the users to directly manipulate a palm rejection region.

II. RELATED WORK

Pen computing usually distinguishes itself through its adaptability that ranges from cell phone devices to tablet PCs, PDAs, electronic whiteboards and surfaces based on projection and multitouch input. Significant advances in mobile technology allow us to research more on pen computing, since current mobile devices have hardware that is much more powerful than early pen computers [2]. One would expect that writing on a tablet with a stylus pen is the same as writing with pen and paper; however, digitizing the pen and paper experience has presented many challenges, which has been the focus of most pen-based computing research. In this section we explore the hardware and software approaches that have been done to address the palm rejection problem.

Hardware-based Solutions

Hardware solutions are mostly reliable in differentiating between a stylus pen input and an unintended finger or palm touch. The *Hand Glider* is a customized glove that covers common parts of the hand that usually triggers unintended touches: the ring finger, little finger, and the side of the palm [4]. Some users prefer to use a stylus pen called an active digitizer pen, such as *Studio Pen* and *iPen*, which transmits the exact coordinate of the touch point on the screen [5,6]. Because it does not rely on the touch capability of capacitive touch screens and more on its ability to send its own location, it automatically rejects any palm touches that will be detected by the display. This makes it much more accurate than a regular rubber-tipped passive stylus pen. Unlike the passive stylus however, active digitizer pens are electronic devices and need a battery to operate. Also, as the active pen transmits its location, some tablet devices will need a separate connector to act as a receiver to determine the active pen's locations. Other devices have active digitizer screens, where the receiver is built in as a screen overlay, such as *Surface Pro* and *Samsung Note* [7,8].

Although hardware-based solutions are accurate and reliable for palm rejection, when these devices are misplaced or lost, the user cannot take advantage of its palm rejection capability. For active digitizer pens, some require their own special software or require a specific platform to run and reliably distinguish touches. In addition, the complexity of these implementations adds to the build cost of the device,

978-1-4799-8660-6/15 $31.00 © 2015 IEEE
DOI 10.1109/NGMAST.2015.45

which makes them usually more expensive and not readily available to all tablet users. Some users opt to look for software applications that have palm rejection support, pairing the software with a cheap passive stylus. The simplicity of the passive stylus makes it easy to obtain and replace unlike active digitizer pens.

Software-based Solutions

Software approaches to palm rejection can be categorized to two major types of implementations: implicit and explicit palm rejection [9].

- **Implicit palm rejection** (Implicit PR) automatically distinguishes errant or unintended palm touches through particular input parameters, without needing the direct or external input from the user. Examples of these input parameters are touch characteristics registered from the device such as contact size, touch area, pressure, and position.
- **Explicit palm rejection** (Explicit PR) defines the palm rejection region by manipulating interactive elements or widgets on the handwriting application. There is usually a draggable area or widget that is used as the palm resting area. Touches that fall in the palm rest area are rejected.

To determine the common type of implementation that is being used in current note-taking applications, we evaluated 20 handwriting applications available from the iTunes App Store, as listed in Table 1 [10]. At the time of our research, ten (10) note-taking applications use explicit palm rejection. These note-taking applications, however, have varying implementations of the palm rejection region.

TABLE I. PALM REJECTION OF NOTE-TAKING APPLICATIONS

Application	Type of Palm Rejection		
	Implicit PR	*Explicit PR*	*None*
Bamboo Paper	✓		
DocAS		✓	
Inkflow		✓	
Inknotes HD		✓	
iNotes		✓	
Mighty Notes Lite		✓	
MyScript Memo			✓
neu.Notes+		✓	
Notability		✓	
NoteBook+ Free			✓
NoteLedge		✓	
Noteshelf		✓	
Paper			✓
PaperDesk Pro			✓
Penultimate	✓		
PhatPad			✓
Rapid Note			✓
Smart Writing Tool – 7notes HD			✓
Tipnote HD			✓
UPAD		✓	

For example, *Notability* features a draggable palm rest area that the user can slide up or down along an axis [11]. *Noteshelf* also implements the same draggable palm rest area as *Notability*, but Noteshelf automatically adjusts after the user has set the initial height of the palm rest area [12]. *NoteLedge*, on the other hand, visually imitates a cloth that can rest under the palm [13]. Unlike Notability that is constrained along one axis, the palm rest area of NoteLedge can move anywhere on the writing area.

We observe that explicit PR implementation is reliable as long as the unintended touches are in the palm rest area. However, most of the rectangular explicit PR implementations work only when the user holds the pen at a higher angle than the hand. Otherwise, the palm rejection area fails to filter touches caused by the upper part of the hand. Explicit PR is also more intrusive than implicit PR, because the user must always take note of the location of the palm rejection region and adjust his pen grip accordingly.

In implicit PR implementations, because the palm rejection area is not displayed, it is more natural for users to write on the display. The user does not have to be constantly mindful of accidentally repositioning the palm rejection area while writing. Implementation of implicit PR is more complex than explicit PR since the algorithm must consider varying grip styles, hand position and handedness. Among the evaluated note-taking applications in Table 1, only two applications, Penultimate and Bamboo Paper, support implicit palm rejection [14,15].

Aside from commercially available software applications, there are also research work that have tackled the palm rejection problem. Shu developed an explicit palm rejection technique called the bezel-focus approach [9]. Users are more inclined to rest their hands on the bezel-focus approach, but some users found it difficult to learn the non-dominant hand interaction, since it is not intuitive or similar to the natural handwriting process. Schwarz et al. developed an implicit palm rejection technique that uses spatiotemporal features, iterative classification and probabilistic voting to determine the writing touch from the palm touches [16]. The system initially shows its best guess to the user, and the guess is refined as more information becomes available. For example, if the touch point is initially assigned as a pen, it draws a stroke on the display. If the guess is later changed, then the guess is removed from the canvas. We refer to this capability as stroke recovery, where implicit PR implementations can try to guess and correct the missed strokes even after the user has done the stroke. As more touch points become available for better algorithm decisions, missed strokes are usually rewritten, while stray marks are removed from the display.

In summary, software solutions to the palm rejection problem include using user-adjustable widgets (explicit PR),

and automatic estimation of hand location based on touch information (implicit PR). Although implicit palm rejection is more intuitive than explicit palm rejection, it is less implemented by software applications because of its inherent complexity and challenges brought about by different handwriting styles.

III. Writing Behavior And Touch Characteristics

Since our goal is to provide users with the closest experience to writing using pen and paper, it is important to explore touch properties and writing behavior while using a stylus pen. Although not explicitly used for palm rejection, Vogel et al.'s study on hand occlusion behavior with direct pen input is useful in understanding writing behavior on tablets [17]. We assume that the occluded area mentioned in their study is similar to the area of our palm rejection region, since that is the area with high probability that users would touch the screen with their hand. Also, we note that the occluded area varies more because of the different pen grip styles instead of anatomical differences. Pen grip styles vary predominantly between three factors: size of the grip, angle of the pen, and grip height. Their experimental results also show that many occluded pixels are higher relative to the pen. This supports our observation that rectangular palm rejection regions from explicit PR cannot filter unintended touches coming from the hand that are located higher than the pen touch.

In addition to hand occlusion behavior, we performed an initial test to verify observations on touch characteristics by other researchers. Users were asked to write naturally on a custom tablet application that logs all detected touches. From the detected touch points, we observed the following touch characteristics:

- The touch points detected from the palm area is a collection of touch points, which often flicker in and out. Contrary to the assumption that the palm generates many touch points once it touches the display, an average of three touch points were observed. Three touch points do not provide enough information to form a complex palm rejection model.
- The touch radius of the stylus or writing touch point is consistent, and having longer and smoother trajectories, while touch points from the palm generally move little, and they vary in size and touch radius.
- Touch radius is a useful parameter in palm rejection design for users who perform writing tasks using the stylus.
- For right-handed users, errant touches rarely occur at the top and left area of the tablet display.

IV. Design of the Palm Rejection Filters

Using the aforementioned observations on touch characteristics and hand occlusion as a starting point for our implementation, we were able to design three touch filters to describe three scenarios:

1.) *First-touch filter*: There are no palm rejection regions defined on the handwriting application. User has not yet touched the display, or the user has lifted his hand from the screen. The algorithm still has to determine the detected touch as a pen or palm input.
2.) *Palm-first filter*: User rested his palm on the display before touching the display with his stylus pen.
3.) *Write-first filter*: User touched the display first using his stylus pen, before resting the palm on the display.

When there is a touch detected from the display (Fig. 1), the algorithm checks whether a palm rejection model already exists. If there is no existing model, the algorithm passes the touch into the First-touch filter, where touch properties, especially the touch radius, determine if the context of the touch is a palm touch or a writing touch. If it is a writing touch, then it is passed on to the write-first filter. If it is a palm touch, then it is passed on to the palm-first filter. If a palm rejection model already exists when a touch is detected, then the algorithm passes the touch to its corresponding filter based on the existing palm rejection model.

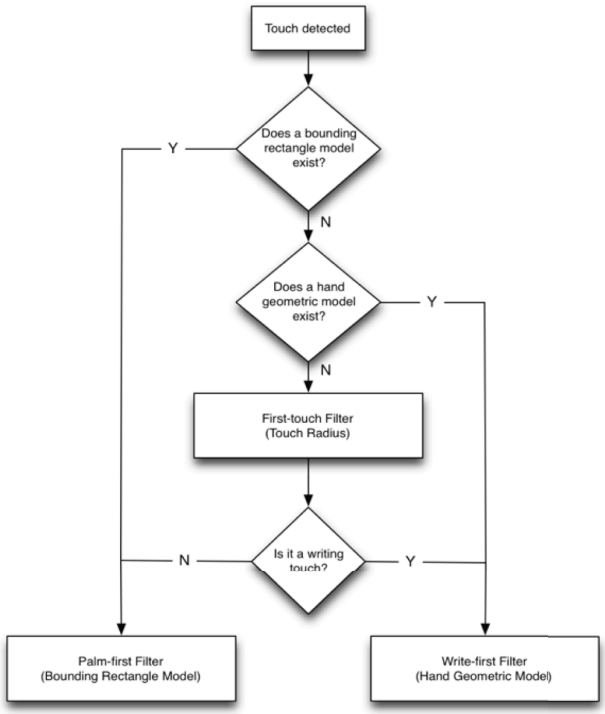


Fig 1. General Flowchart

A. First-Touch Filter

If the palm rejection region is not yet defined, we rely solely on the touch characteristics to determine the context of the touch, whether the touch is the writing touch or the palm touch. We use the following touch properties to filter the touches: touch radius, touch phase and touch location.

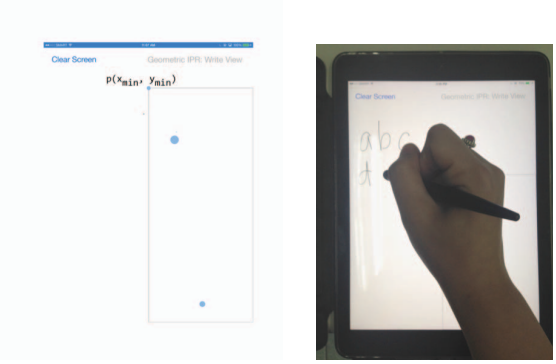
If the detected touch is less than the mean touch radius, then we assume it is the intended writing touch and we send the

touch to the Write-first filter. If it is greater than the mean touch radius, then it is a palm touch, and we send the touch to the Palm-first filter.

B. *Palm-First Filter*

As mentioned in Section 3, resting the palm on the display does not instantly provide enough touch points to form a complex palm rejection model. Instead, we use a simpler palm rejection model, the bounding rectangle model. Some implementations implicitly use the largest bounding box that encompasses detected touch points, expanding the box with a nominal value to "predict" other near touch points. However, this does not take into account the touch point from the bone of the wrist. We extend the bounding rectangle model to the edge of the screen, to reject palm touch points that may be generated from the arm as well.

When the palm is rested on the display before the writing touch, the large touch radius obtained from the palm is stored in a palm touch array. From this palm touch array, we obtain the x_{min} (the leftmost x-coordinate) and y_{min} (the topmost y-coordinate) to define the corner of the bounding rectangle model that we use for the palm rejection region (Figure 2a). We use the touch phase to add and remove palm touches from the array, so that the bounding rectangle will adjust accordingly as touches begin and end. We show in Figure 2a the bounding rectangle model on our handwriting application, with the blue points showing the detected palm touch points being encompassed by the model. Figure 2b shows the bounding rectangle model in action when the palm is first rested on the screen before writing.



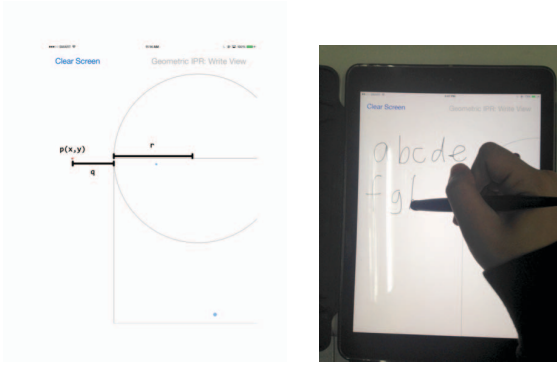
(A) (B)

FIG 2. PALM-FIRST FILTER

C. *Write-first Filter*

Once the writing touch has been determined from First-touch and Palm-first filters, we would have enough touch information to form a better hand model filter. We then switch the palm rejection model to the hand geometric model. The hand geometric model is loosely based on Vogel et al's 5-parameter geometric model, which we modified to reduce complexity of the geometric model [17]. We used a rectangle, to catch varying degrees of the hand/arm pivot, and an arc (semi-circle) to catch the shape of the fist. We define the following parameters shown in Fig 3a: *q* is the distance from the touch point to the edge of the arc while *r* is the radius of the arc. Although meant as a rough guide for designers for occlusion-aware interfaces, we used the mean values from Vogel et al's geometric model as the base mean values for our own model to test whether these can also be effectively used for palm rejection.

To accommodate the range of grip styles, we automatically adjust the distance q by getting information from our palm touch array. We compute the distance between the writing touch point p(x,y) to the $p(x_{min}, y_{min})$ coordinate of the bounding rectangle model, set as q_{br}. If we find that q_{br} less than the distance q, which means that the grip is tighter, we use q_{br} as the new distance q. When the user lifts his hand on the display, we set q to its original base value. We show in Figure 3a the hand geometric model on our handwriting application, while Figure 3b shows the hand geometric model in action when the stylus pen first touches the display before the palm.



(A) (B)

FIG 3. WRITE-FIRST FILTER

V. EXPERIMENTAL RESULTS

To determine the performance of our palm rejection algorithm, we compared our note-taking applications to commercially available handwriting applications with implicit palm rejection. From Table 1 in Section 2, only two note-taking applications have implicit palm rejection capability: Penultimate and Bamboo Paper. Fifteen (15) participants, all right-handed, were asked to do the writing tasks on the iPad Air. The ages of the participants range from 20-36 years old, with a mean age of 25. Prior to testing, all of the participants have already used a tablet device, twelve (12) participants have used a stylus, and nine (9) participants have used a handwriting application on a tablet.

Users were provided a passive stylus pen to do the writing tasks. The first writing task (Fig. 4a) was to write the English alphabet in uppercase letters along with the numbers 0 - 9. For the second and third writing tasks (Figs. 4b and 4c), users were instructed to write the known pangram, "The quick brown fox jumps over the lazy dog", in uppercase and lowercase respectively. We have opted to test the English alphabet as all of the test participants are already familiar with it, and they will be able to write the alphabet as natural as possible in their own handwriting.

(A) (B) (C)

FIG 4. WRITING TASKS

The first writing task aims to get more defined strokes from the participants. The second and third writing tasks aim to check the ability of the palm rejection algorithms on regular sentence writing, while still covering all of the letters in the English alphabet. We encouraged participants to write naturally as if they were writing on paper, and were advised that they were free to rest or reposition their hand anywhere on the screen. They were also instructed to continue writing even if they see no writing stroke on the display. The writing tasks were repeated for three handwriting applications -- our hand model-based filter handwriting application (model-based IPR), Penultimate and Bamboo Paper.

The experiment was set up as follows: a camera is placed on level height with the tablet such that the camera captures when the stylus pen is touching the tablet. The tablet display is also simultaneously streamed with the video camera in a different display monitor, so that we can see strokes that were occluded by the participant's hand. Fig. 5 shows a screenshot of the video and note-taking application stream during one of our tests.

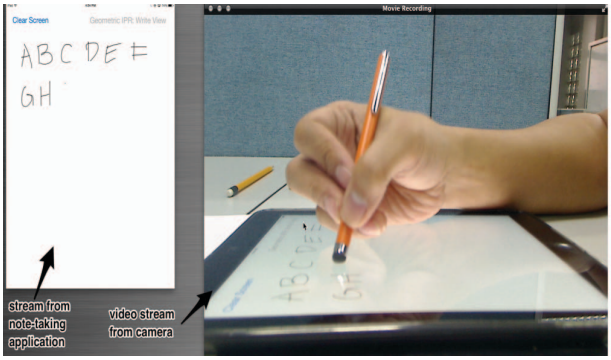


FIG 5. EXPERIMENT SETUP

Through video analysis and manual annotation, we recorded the number of strokes that each participant has drawn during the writing tasks. We chose to collect and analyze the participant's writing strokes instead of palm data, since it is not possible to collect palm touch information in Penultimate and Bamboo Paper. We categorized the writing strokes as follows: correctly written strokes were counted as true positives, missed strokes were counted as false negatives, while strokes that appeared on the display but did not come from the stylus pen (unintended touches) were counted as false positives. True positive and false negative strokes are the true writing strokes done by the user, while false positives are extraneous or unintended strokes caused by the palm or the other parts of the hand. The writing accuracy of the system is determined by the following equation:

$$writing\ accuracy = \frac{true\ positive}{true\ positive + false\ negative} \quad (1)$$

We determine the errant touch rate by calculating the number of unintended touches over the total true writing strokes, as shown in the following equation:

$$errant\ touch\ rate = \frac{false\ positive}{true\ positive + false\ negative} \quad (2)$$

At the end of the writing tasks, each participant rated each application's comfort and accuracy, using the following scale:

1- Very Poor 2 - Poor 3-Fair 4-Good 5-Very Good

The Comfort criterion pertains to the naturalness of the participant's writing on the tablet device using the application, especially if they were able to rest and move their hand on the tablet screen comfortably while writing on the tablet. The Accuracy criterion is the participant's perceived ability of the application to reject errant touches and its consistency in rejecting palm touches; participant should consider if he was able to write consistently during the testing, or if the application fails to recognize his writing strokes.

VI. RESULTS AND DISCUSSION

Table 2 summarizes the cumulative writing strokes for the 15 participants. Since Penultimate has stroke recovery capability, we also considered Penultimate data both before and after the writing strokes were corrected by its stroke recovery algorithm. Initial analysis show that Penultimate was able to recover most of the missed strokes and stray marks. As shown in Table 3, our model-based IPR has a higher writing accuracy (99.19%) compared to Penultimate before stroke recovery (98.81%) and Bamboo Paper (82.16%). Although the writing accuracy of Penultimate after stroke recovery (99.34%) is higher than our model-based IPR, our model-based IPR still obtained a lowest errant touch rate than both Penultimate and Bamboo Paper at 0.87%. This means that although writing accuracy is comparable to Penultimate, our model-based IPR is still less prone to stray marks than the other two note-taking applications.

TABLE II. SUMMARY OF WRITING STROKES

	Total Writing Strokes	*True Positive (correct strokes)*	*False Negative (missed strokes)*	*False Positive (stray marks)*
1. Penultimate	3357	3317	40	656
2. Penultimate with Stroke Recovery	3339	3317	22	210
3. Bamboo Paper	3335	2740	595	759
4. Model-based IPR	3342	3315	27	29

TABLE III. WRITING ACCURACY

	Writing Accuracy	*Errant Touch Rate*
Penultimate	98.81%	19.54%
Penultimate (with Stroke Recovery)	99.34%	6.29%
Bamboo Paper	82.16%	22.76%
Model-based IPR	99.19%	0.87%

Qualitative tests on the implemented algorithm included criteria rating on its comfort and accuracy. The mean criteria rating were computed based from the participants' rating as shown in Table 4. Results show that all three application range from 'Fair' to 'Good', which means the participants still found it comfortable to write on the tablet display. Bamboo Paper, however, obtained a 'Poor' mean criteria rating for accuracy, while the two other participants obtained a 'Fair' to 'Good' criteria rating for accuracy. Our model-based IPR obtained the highest criteria rating for both accuracy and comfort among the three note-taking applications.

TABLE IV. CRITERIA RATING

	Comfort	*Accuracy*
Penultimate	3.9	3.4
Bamboo Paper	3.5	2.2
Model-based IPR	4.3	4.2

Lastly, the test participants were also asked to choose their preferred note-taking application based on their palm rejection capability. Nine (9) out of 15 participants chose our model-based IPR, two participants chose Penultimate, while four participants chose Bamboo Paper. It is interesting to note that although Penultimate has a higher writing accuracy, more participants still chose Bamboo Paper. Although this is mostly the participant's preference and will depend on varying factors beyond the scope of our research (such as UI elements and pen stroke quality), we also observed that during the tests, participants were initially confused when they see displayed strokes on supposedly missed or blank strokes, or see disappearing stray marks way after the writing stroke had been written. We surmise that stroke recovery, although more accurate, is an unusual behavior if compared to traditional handwriting. This reinforces our need to provide a real-time palm rejection algorithm.

VII. CONCLUSION AND RECOMMENDATIONS

In this research, we designed a real-time model-based implicit palm rejection algorithm based on hand occlusion behavior and touch characteristics. We show the effectiveness of our algorithm through a series of writing tasks with two commercially available note-taking applications with implicit palm rejection capability. Results show that our model-based IPR has a writing accuracy that is comparable to current note-taking applications, but provides a better errant touch rate.

Our model-based IPR algorithm is currently modeled for right-handed individuals, while our tests focused more on the effectiveness and naturalness of handwriting using our model-based IPR. Future work includes further tests on the algorithm's effectiveness in sketching, and the design of additional filters to support left-handed individuals.

ACKNOWLEDGMENT

We thank the ERDT program under the DOST-SEI for the scholarship and funding support of this research.

REFERENCES

[1] R. Zeleznik and T. Miller, "Applications and issues in pen-centric computing" in *IEEE Ann. History of Computing*, 2008, pp. 14–21.

[2] K. Hinckley et al. "Pen + touch = new tools" in *Proc. 23rd Annu. ACM symposium on User interface software and technology - UIST '10*. 2010, pp. 27.

[3] M.J. Camilleri et al. "Touch displays: the effects of palm rejection technology on productivity, comfort, biomechanics and positioning." in *Ergonomics*, Vol. 56, 2013. pp. 1850–62. doi:10.1080/00140139.2013.847211

[4] Alamont Design LLC. (2011). *The Hand Glider*. [Online]. Available: http://www.thehandglider.com/aboutus.asp.

[5] Byzero Inc. (2012). *Studio Pen*. [Online]. Available: http://www.by-zero.com/?page=studiopen.

[6] Cregle Inc. (2014). *Cregle iPen Stylus*. [Online]. Available: http://www.cregle.com

[7] M. Crider, (2011). *Samsung highlights the Galaxy Note's Wacom digitizer*. [Online]. Available: http://androidcommunity.com/samsung-highlights-the-galaxy-notes-wacom-digitizer-20111027/

[8] Microsoft. (2014). *Surface Pro 2*. [Online]. Available: https://www.microsoft.com/surface/en-us/products/surface-pro-2/

[9] K. Shu, "Understanding and Rejecting Errant Touches on Multi-touch Tablets". MS. Thesis, Singapore Management Univ. 2013.

[10] Apple. (2014). *Apple App Store.* [Online]. Available: http://itunes.apple.com.

[11] Ginger Labs. (2014). *Notability*. [Online] Available: http://www.gingerlabs.com

[12] Fluid Touch. (2014). *Noteshelf*. [Online] Available: http://www.noteshelf.net

[13] Kdan Mobile. (2014). *NoteLedge.* [Online] Available: http://www.kdanmobile.com/en/noteledge

[14] Wacom. (2014) *Bamboo Paper*. [Online] Available: http://bamboopaper.wacom.com.

[15] Evernote. (2014) *Penultimate*. [Online] Available: http://evernote.com/penultimate.

[16] J. Schwarz et al. "Probabilistic palm rejection using spatiotemporal touch features and iterative classification" in *Proc. SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). New York, NY, USA, ACM, 2014. doi:10.1145/2556288.2557056

[17] Vogel et al. "Hand occlusion with tablet-sized direct pen input." in *Proc. 27th International Conference on Human Factors in Computing Systems - CHI 09*, (c), 557. doi:10.1145/1518701.1518787