Exploring and Understanding Unintended Touch during Direct Pen Interaction

MICHELLE ANNETT, University of Alberta, Microsoft Research
ANOOP GUPTA, Microsoft Research
WALTER F. BISCHOF, University of Alberta

The user experience on tablets that support both touch and styli is less than ideal, due in large part to the problem of *unintended touch* or *palm rejection*. Devices are often unable to distinguish between *intended touch* (i.e., interaction on the screen intended for action) and *unintended touch* (i.e., incidental interaction from the palm, forearm, or fingers). This often results in stray ink strokes and accidental navigation, frustrating users. We present a data collection experiment where participants performed inking tasks, and where natural tablet and stylus behaviors were observed and analyzed from both digitizer and behavioral perspectives. An analysis and comparison of novel and existing unintended touch algorithms revealed that the use of stylus information can greatly reduce unintended touch. Our analysis also revealed many natural stylus behaviors that influence unintended touch, underscoring the importance of application and ecosystem demands, and providing many avenues for future research and technological advancement.

Categories and Subject Descriptors: H.1.2 [**User/Machine Systems (e.g., Human Factors)**]; H.5.2 [**User Interfaces (e.g., Mouse, Touchscreen)**]; H.5.m [**Information Interfaces and Presentation (e.g., HCI)**]: Miscellaneous

General Terms: Design, Human Factors, Measurement

Additional Key Words and Phrases: Unintended touch, intended touch, palm recognition, palm rejection, pen, stylus, tablet, hover, rejection regions, direct interaction

ACM Reference Format:
Michelle Annett, Anoop Gupta, and Walter F. Bischof. 2014. Exploring and understanding unintended touch during direct pen interaction. ACM Trans. Comput.-Hum. Interact. 21, 5, Article 28 (November 2014), 39 pages.
DOI: http://dx.doi.org/10.1145/2674915

1. INTRODUCTION

The world of touch interaction today is largely focused on direct manipulation, mapping users' taps, swipes, and gestures to selection, panning and zooming. Such mappings allow natural interactions, with all touch events being intended for interaction. A similar situation is found with stylus-only devices, where all input is mapped to inking, assuming that every stroke is intentional. This enables users to easily and fluidly complete tasks that require enhanced precision, such as, for example, diagramming, sketching, and writing. The combination of pen and touch is incredibly powerful, enabling the dominant hand to perform inking and the nondominant hand to provide support via manipulation [Guiard 1987; Hinckley et al. 2010a, 2010b]. Such a combination allows

Authors' addresses: M. Annett and W. F. Bischof, Department of Computing Science, 2-32 Athabasca Hall, University of Alberta, Edmonton, Alberta, Canada, T6G2E8; emails: {mkannett, wfb}@ualberta.ca; A. Gupta, One Microsoft Way, Redmond, Washington, United States of America, 98052; email: anoop@microsoft.com.

DOI: http://dx.doi.org/10.1145/2674915

a seamless transfer of natural and intuitive behaviors from the physical to the digital world while also supporting the implementation of new tools and interaction techniques [Fitzmaurice et al. 1999; Hinckley et al. 2010a, 2010b, 2013; Vogel et al. 2009; Wagner et al. 2012; Yoon et al. 2013; Zhang et al. 2012].

Such interaction paradigms require that the system not only distinguishes between pen and touch input but also distinguish between touch events that are intentional and those that are an unintended by-product of stylus-based interaction. While annotating a document or writing notes, it is common to rest the palm, wrist, or forearm on the screen for support, to reduce fatigue, or to anchor the page [Hancock and Booth 2004; Siio and Tsujita 2006]. This natural behavior unintentionally generates a variety of touch input events. Depending on the digitizer, operating system, and application, such events may invoke a gesture, manipulate content, or render markings on the screen [Annett et al. 2014b]. When using these systems, there are many instances where the user would like the touch input to be recognized (e.g., while swiping to the next page or zooming an image). Such scenarios exemplify one of the major problems plaguing pen and touch interaction today: How does a system determine which touch events are *intended* (i.e., intentional interactions with the fingers) and those that are *unintended* (i.e., those that are a by-product of the skin resting on, or grazing the display)?

The identification and rejection of unintended touch events is important for all pen and touch-enabled devices, and especially so for tablets. During simultaneous bimanual manipulation, that is, when actively using both pen and touch (e.g., Hinckley et al. [2010a, 2010b]), the touch events corresponding to the gestures need to be accepted, whereas the touch events generated by resting the stylus hand on the screen should be ignored. With interleaved pen and touch, where only one modality is being "used" at a time, the system must still distinguish whether a touch is intended for action or if it is simply the arm contacting the screen prior to the stylus being placed down. Failure to properly identify the intention of the touch results in spurious input to the application, severely reducing a user's efficiency and satisfaction with an application or device [Annett et al. 2014b].

The need to accurately distinguish between intended and unintended touch input has been identified extensively in prior work [Annett et al. 2014b; Alcantara et al. 2013; Gerken et al. 2010; Hinckley et al. 2010; Lin et al. 2013; Matulic and Norrie 2012; Vogel and Balakrishnan 2010a; Valderrama Bahamonde et al. 2013; Zelzenik et al. 2008] but has yet to be explored in depth. Most developers and researchers today ignore the problem completely by turning touch off or applying existing work from occlusion-aware interfaces to the problem [Yoon et al. 2013]. There is a lack of knowledge both in the literature and in practice regarding unintended touch and appropriate interventions.

Apart from drawing attention to the importance of understanding and solving unintended touch, this work contributes to the understanding of the problem itself, detailing specific challenges developers and manufacturers are faced with when devising solutions to unintended touch. The current state of the art, in terms of unintended touch algorithms that exist in industry and in research, is then explored. Motivated by the diverse approaches that have been tried, an experiment is described that gathered motion capture data from a stylus and tablet and raw data from a touch and stylus digitizer while participants were writing, drawing, and annotating documents. The dataset of unfiltered and unprocessed touch data was recorded directly from the digitizer and permitted an evaluation and comparison of novel and existing solutions to unintended touch, given both the technological capabilities today and those likely in the near future. A number of natural user behaviors that influenced unintended touch requirements and algorithm success are also detailed. The results from this exploration and the behavioral patterns identified have been synthesized into two broad areas for

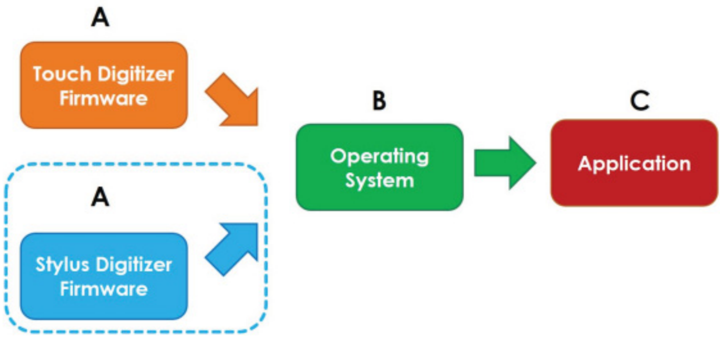


Fig. 1. The data pipeline from the digitizer to an application. Rejection can be performed at three stages of the pipeline: (a) within the firmware of the device, (b) within the operating system, and (c) within individual applications. Depending on the device, the stylus digitizer may be absent (i.e., capacitive devices) or integrated with the touch digitizer.

future work that should be of great interest to the pen computing community and stimulate the stylus ecosystem for many years to come.

2. UNDERSTANDING UNINTENDED TOUCH

Within industry, the problem of unintended touch is often termed *palm rejection*. By its very description and name, palm rejection suggests the need to reject only the palm. As the exploration here illustrates, users are likely to touch the display using fingers, knuckles, palms, wrists, and forearms while inking. The presence and dispersion of touch data found across the screen suggests that "palm rejection" must not focus exclusively on identifying and ignoring the palm. Researchers and industry practitioners should thus use more appropriately descriptive terminology, such as unintended touch, moving forward. Framing the problem as unintended touch, instead of palm rejection, inherently implies that all future approaches and conversations consider all the skin surfaces that can touch the device, not just the palm. This will go a long way in drawing attention to the importance of the problem and prevent common comments such as "oh that's an easy problem—just reject everything larger than the finger." As demonstrated, approaches based on these off-the-cuff comments are inadequate.

Determining if a touch event is intended or unintended is a multifaceted problem influenced by many factors. These factors have been distilled into three main categories: time and data available for decision making, the types of input that need to be disambiguated, and the cost of recovering from incorrect rejection.

2.1. Timing and Data Available for Unintended Touch

The rejection of unintended touch can be performed at three stages along the data pipeline [Hinckley and Wigdor 2012]: in the firmware, in the operating system, or in the application (Figure 1). Each stage has different time constraints and different data available to make rejection decisions.

Solutions to unintended touch at the firmware level have full access to raw, unfiltered data. The raw data is the least abstracted and most informative available along the pipeline and its usage calls for primitive rejection solutions involving input shape, simple spatial heuristics, or raw sensor magnitude values. Rejection performed at this stage does not allow per-user personalization or the integration of factors such as task context, as the information being passed through the digitizer's controllers to the operating system is already close to or at bus capacity. Developers can, however, ensure a consistent user experience across operating systems and devices with the same digitizers and firmware provided that rejection is performed at this stage. The caveat, of course, is that decisions need to be made very quickly so as to prevent introducing latency into the entire system. Through consultation with industry practitioners, hardware designers, and manufacturers, it was understood that there is often less than 5ms allotted to collect and process any necessary data and make a rejection decision.

Unintended touch algorithms at the operating system level can provide a consistent user experience across all applications on the same platform but run the risk of inhibiting or eliminating certain interaction techniques or domain-specific functionality, such as simultaneous or interleaved pen and touch input [Hinckley et al. 2010a, 2010b]. At this level, solutions have access to processed and simplified data from the digitizer and can harness user-specific information such as handedness, hand posture, and so on. Time requirements at this stage are similar to those at the firmware stage, as any delay propagates to all applications. With the Windows 7 operating system, for example, all touch events are rejected when the pen is detected in the hover state [Buxton 1990], resulting in consistent but restrictive pen and touch experiences across all applications. Similar to the firmware approaches, there is very little time allotted to make decisions, typically less than 5ms.

Solutions to unintended touch at the application level can use personalized information about handedness or hand posture along with any domain or application-specific knowledge. Any touch information available to developers, however, is abstracted: one does not have access to anything more than the location and radius or bounding box of a touch input that is provided by the operating system. The use of application-specific rejection also introduces inconsistency within the application ecosystem, resulting in confusion when rejection fails. Development costs for all applications increase by forcing the decision to the application stage, as solutions must be written for each application. The length of time available to make rejection decisions is limited only by the amount of latency developers want to inject into their applications. Each application can weigh the cost of accuracy against the cost of adding increasing latency. As latency perception appears to be task dependent, and dependent on the demands of the application, more or less time may be available for rejection [Annett et al. 2014c; Ng et al. 2014].

When evaluating and understanding unintended touch, it is imperative to consider not only the accuracy of an approach or its data requirements but also the duration of time that is available to make an "accept" or "reject" decision. Regardless of when rejection is performed, the data that needs to be collected and processed for rejection, in addition to the actual rejection decision, should be handled in the most efficient manner possible. Participants in the user studies conducted by Annett et al. [2014c] and Ng et al. [2014] noticed—and complained about—the lag found with stylus-enabled devices today; hence, any additional lag introduced by the rejection process and its algorithms could degrade the user experience further [Laundry 2011; Ng et al. 2012].

2.2. Input to Be Rejected

One of the most difficult aspects of unintended touch concerns the variety of touch inputs received from the digitizer. Many consider the problem restricted to the palm, as evidenced by the frequent terming of the problem as "palm rejection" [Zeleznik et al. 2008]. On capacitive touch-only devices, however, any skin contact with the screen can initiate a touch event. Whenever the knuckles, wrist, palm, fingers, forearm, and so on touch the screen, a touch event is generated (Figure 2, and further Figures 14, 20, 24, and 26). As these devices do not have dedicated stylus digitizers, the stylus is undistinguishable from the fingers, indicating that the stylus first needs to be disambiguated from touch before each touch event can be labeled as either intended or unintended. With devices that contain NTrig or Wacom digitizers (e.g., Wacom Cintiq, Samsung Business Slates, and Surface Pro), distinguishing between pen and touch is not an

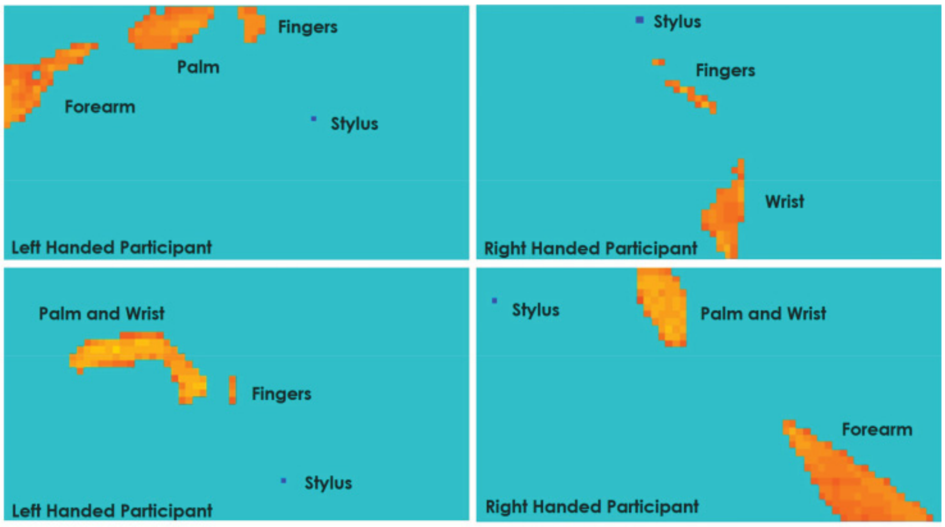


Fig. 2. Example data provided by pen and touch digitizers. The stylus location is represented in dark blue and the touch data is represented in orange, with the darker shades representing the higher levels of activation. All touch data in these images is unintended and should not be accepted.

issue. However, the disambiguation between unintended and intended touch still remains. As the skin can touch the screen at any time and in any location, determining the intentionality of touch is still very difficult, even with an active stylus.

Although it may seem as if unintended touch is a “problem” that needs to be solved, it is also important to consider the identification of the palm, fingers, knuckles, and forearm as an opportunity for novel interaction, gesturing, and so on. In earlier work by Cao et al. [2008], Marquardt et al. [2011], Widgor et al. [2011], Wu and Balakrishnan [2003], and Yan et al. [2013], the classification of a touch input as the palm, a fist, or the side of the hand, was harnessed to create novel gestures for 2D and 3D content manipulation. Brandl et al. [2009], used palm identification to correctly orient occlusion-aware user interaction menus and elements toward the user. In other work, such information was instrumental in recovering the identity of users and their touch points, while they interacted around a multitouch tabletop [Annett et al. 2011; Murugappa et al. 2012; Ramakers et al. 2012]. One could also imagine many other scenarios that could capitalize on the identification of various hand parts (e.g., multipen input, technology-assisted rehabilitation, content occlusion). Given the multitude of possibilities outside unintended touch, we thus encourage the research community to continue exhausting and exploring the usefulness of hand-part identification.

2.3. Cost of Recovery from Incorrect Rejection

It is imperative to consider the cost of imperfect rejection. There is a delicate balance between being overly cautious (i.e., falsely rejecting too many intended touch events) and overly optimistic (i.e., falsely accepting too many unintended touch events). If the user can easily recover from the errors introduced by poor rejection (e.g., when touch causes a document to scroll prematurely, when there is an easily accessible undo button, or when the user can flip the stylus over and use the eraser), then it may be beneficial to be optimistic about touch events. In contrast, if the cost of recovering from errors is high (e.g., when the user has to switch modes to erase a stroke precisely or when undoing a series of manipulations), then a cautious approach may be best. It is thus essential for developers and designers to consider the cost of recovering from errors when deciding about the functionality associated with pen and touch and when making decisions about unintended touch.

3. CURRENT STATE OF THE ART

Current approaches to unintended touch are commonly spurred by the desire of application developers to implement solutions that mesh with their end-user population and interaction schemas. In instances where solutions have yet to be implemented, there are several user-specific behaviors that have been used to fill the void. The following review of current solutions to unintended touch is not meant to be exhaustive, given that numerous applications are continually updated with novel or improved solutions every week. Instead, an overview of the range of possibilities that have been used is provided.

3.1. User Adaptations

When touch rejection algorithms are unavailable in applications, users tend to make behavioral adaptations. By holding the hand, wrist, or forearm aloft instead of resting them on the screen, users can reduce the amount of incidental contact with the screen [Annett et al. 2014b; Gerken et al. 2010; Vogel and Balakrishnan 2010a]. Since the lower arm is not stabilized, users experience increased fatigue, decreased pointing accuracy [Matulic and Norrie 2012], decreased legibility of notes, diagrams, or sketches, and increased frustration [Annett 2014a; Annett et al. 2014b]. Some users have resorted to wearing a nonconductive glove on their dominant hand to prevent unintended touch [SmudgeGuard, n.d.]. The use of such gloves negates the need to alter the hand or wrist posture but changes the friction profile of the palm against the screen and adds overhead, as the user has to put on and take off the glove between uses.

3.2. Firmware Approaches

Recent advances in signal processing have enabled manufacturers to better disambiguate between pen and touch. By increasing signal to noise ratios, Atmel [2014], Synaptics [Coldewey 2011; Sage 2011], Samsung [SmartKeitai 2013], and many others have made it easier to differentiate between narrow activation signals generated by thin, pointed objects such as styli, and larger, bulbous objects such as fingers, palms, wrists, or forearms. These prototypes have eliminated disambiguation on capacitive touchscreens but have yet to solve unintended touch, as the intention of touch-based events is unknown.

It is difficult to determine if any firmware-specific approaches have been implemented, as the average developer only has access to abstracted data available from the operating system. It has been reported, however, that the Microsoft Surface Pro has an unintended touch algorithm in the touch and stylus controller firmware (i.e., Palm Block Technology), but few details are available regarding the nature of the implementation [Microsoft 2013]. Regardless of the algorithms employed by the Surface Pro, the resulting data is propagated to the operating system and combined with the additional unintended touch approaches found in the Windows operating system, as well as any approaches that specific applications may use.

3.3. Operating System Approaches

To date, there is one approach to unintended touch that has been implemented at the operating system stage. On Windows 7 and 8 devices, whenever the stylus is in the hover state [Buxton 1990], all touch events are considered unintended and rejected. This prevents simultaneous interaction from occurring whenever the stylus is within range. Such an approach requires a digitizer that detects hover (or a Bluetooth-enabled stylus reporting hover) but is challenged by the nonlinearity of the digitizer and variety of hover heights found on devices. In Windows 8.1, for example, a hover height of 20mm is required for logo certification [Microsoft 2014], but even at this height, many incidental contacts are still processed. The present evaluation takes these limitations into account and determines the hover height necessary for this approach to be successful from the behavioral and digitizer perspectives.

3.4. Application-Specific Approaches

Most unintended touches are generated by areas of the skin that are substantially larger than the fingers (e.g., the forearm or the palm). Applications such as the Bamboo Paper on the iPad capitalize on the different-sized areas that the skin activates on the digitizer and ignores areas above a certain threshold, similar to what Murugappana et al. [2012] proposed for multitouch tabletops. Although simple, such an approach requires that input processing must be delayed because all contacts initially appear very small on the digitizer, making it virtually impossible to distinguish the palm from the finger on initial contact. If some latency is acceptable, this approach can work well, but it fails if a finger-sized contact (e.g., the knuckle or fingertip) unintentionally touches the screen. The present analysis focuses specifically on the usefulness of contact size for rejection, identifying the appropriate threshold for rejection and understanding the type of touch inputs and use cases that could cause such an approach to fail.

When access to raw touch data is not available, developers sometimes employ simple models based on where the palm is hypothesized to rest while taking notes (e.g., the bottom of the screen). These static “rejection regions” (i.e., areas of the screen where all touch input is ignored) are used to reject unintended touch quickly and with little processing overhead. In the Moleskin Journal iPad application, for example, a static horizontal widget persists in the bottom third of the screen, rejecting all touch input. In Samsung’s S Note application and in OneNote 2013, the rejection region is extended such that it covers the whole inking canvas but not the menu or toolbars. A dedicated menu button allows users to toggle touch input on and off, which allows the pen to ink exclusively. Shu [2013] proposed a variant of this approach, allowing the user to specify the location of the “safe” input region by tapping near the bezel with the nondominant hand and manipulating the region’s size using a “zoom in” gesture. Coupling a visual representation (e.g., a colored region or a menu button) with rejection allows users to understand where they can rest their palm, decreasing the likelihood of unintended touch. These approaches, however, prohibit efficient simultaneous and interleaved pen and touch interaction and cannot accommodate rotation of the hand or tablet while sketching [Fitzmaurice et al. 1999]. As there are many ways to define and place rejection regions on the screen, the present work explores a variety of possible regions using static heuristics similar to those of the Moleskin application as well as novel dynamic parameters such as the location of the stylus. This extensive evaluation leads to a deeper understanding of rejection regions and to solutions that harness generalized information. The design of feedback to alert users about safe and unsafe regions of the screen is left for future work.

Handedness information or models of common hand postures or touch input can also be used to improve rejection algorithms. Vogel et al. [2009] and Vogel and Balakrishnan [2010a] proposed a general geometric model of hand occlusion as a method for eliminating problems associated with occluded screen content. In their model, the hand and wrist are generalized as a circle and rectangle whose location and size are based on the aggregated video data from a sample of users. In TextTearing, Yoon et al. [2013] have recently used the model to overcome issues with unintended touch. A similar approach is used by the Penultimate iPad application, which requires users to specify which of three models of hand posture is most similar to their own. It then uses this information to reject all touch events that occur in a hand-like shape to the right (or left) of the stylus. As Annett et al. [2014b] demonstrated, there are many possible hand postures and grips while inking and interacting with pen-enabled devices, the Vogel approach was included in the study as an example of the narrowest possible dynamic rejection region. Recently, Schwartz et al. [2014] proposed the use of a spatiotemporal approach that harnesses the radius of each touch event to determine if it should be rejected. With their approach, the size of each touch input was compared to a dataset and machine learning techniques were applied for rejection resulting in the acceptance of 98% of all touch events. While this seems impressive, such results come at the cost of delaying the ink strokes that were presented to users by upwards of 100ms. As previously identified, the introduction of any latency into the pipeline will be noticed by users and influence their behavior, making such an approach infeasible given the strict requirements laid out in Section 2.

4. DATA COLLECTION

To better understand natural skin contact with pen- and touch-enabled devices, an experiment was conducted to gather motion capture and raw digitizer data while users were inking. This data provides information on natural inking behavior and forms a basis for the analysis of approaches to unintended touch.

4.1. Participants

Eighteen participants (three female) were recruited from within Microsoft Research to participate in the study (M = 42 years, SD = 10 years, range = 25–61 years). Participants were naïve to the purpose of the experiment and were not aware that unintended touch was being evaluating to ensure that natural behavior, and not adaptations were being recorded. Four participants were left-handed, as assessed by the Edinburgh Handiness Inventory [Oldfield 1971]. Participants had varied experience using a stylus with a tablet, with some using a stylus and tablet every day, others only a tablet, and a few having never used a stylus or tablet. All participants received a $10 honorarium at the conclusion of the approximately 30-minute experiment.

4.2. Apparatus and Equipment

Participants were seated at a desk surrounded by a six-camera Optitrack motion capture system and three Microsoft LifeCam Cinema web cameras. The Optitrack system recorded the movement of the stylus around and above the tablet surface as well as the movement of the tablet itself. A custom C++ application recorded the location of the stylus and tablet at 100Hz, with ±1mm of error. The three web cameras were placed in front, to the right, and to the left of the interaction volume to record overt participant behavior. The cameras recorded the complete interaction volume with 1080p resolution at 30 frames per second (Figure 3). The iSpy open-source software was used to synchronize, timestamp, and record the web cameras for later offline processing and analysis.

Participants were provided with a Sony Vaio Duo 11 tablet that utilized an NTrig touch and stylus digitizer. Plastic blocks of one centimeter height were affixed to the bottom of the tablet to ensure that the markers on the tablet would not be disturbed if the tablet was moved. During initial pilot testing of the hardware, software, and tasks, the blocks were not found to influence the size, shape, or location of touch and stylus input that was generated. Five motion capture markers were added to the tablet, allowing its position and orientation to be recorded (Figures 4(a) and 4(b)). The tablet was oriented in landscape mode and participants were free to rotate or move the tablet but were instructed to keep it on the table and within the orange interaction area.

An active NTrig stylus was provided and participants were instructed to hold it in their dominant hand (Figure 4(c)). The stylus was modified by adding three markers

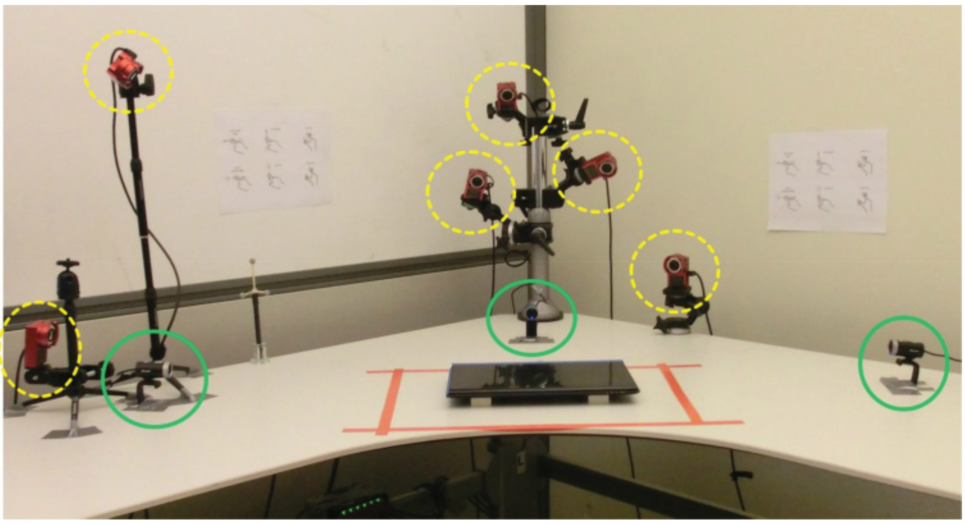


Fig. 3. The experimental setup with the six-camera Optitrack system (highlighted with yellow dashed circles) and three web cameras (highlighted via green solid circles). The interaction area was marked off with orange tape.

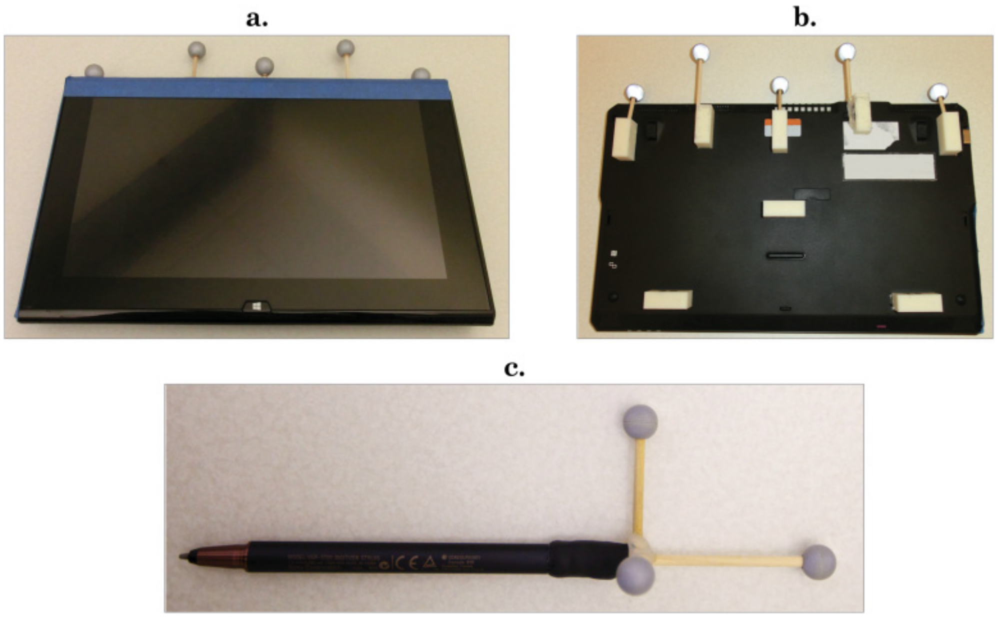


Fig. 4. The motion capture markers attached to the (a) the front of the Sony Vaio Duo 11 tablet with the motion capture markers, (b) the back of the tablet with small blocks were affixed to ensure that the motion capture markers were not moved or damaged, and (c) the Ntrig stylus.

on the back to track its position and orientation. For calibration, a fourth marker was added to the tip of the stylus, allowing the position of the stylus tip to be extrapolated from the remaining three markers during the experiment. The markers added 3g to the weight and 88mm to the length of the stylus.

The tablet had a resolution of 1,920 × 1,080 and ran Windows 8. The tablet also ran a custom C# and WPF inking program that recorded all touch and stylus input from the 64 × 36 sensor array as raw, unfiltered antenna magnitudes. Each individual sensor detected a contact area of approximately 16mm^2. The sensor magnitudes came directly from the digitizer via specialized software provided by NTrig and were converted into raw touch and stylus data. The data was recorded at 60Hz and bypassed all Windows-specific formatting, filtering, and rejection approaches. Each ink stroke was rendered at 60 frames per second, due to the display refresh rate of the tablet's screen.

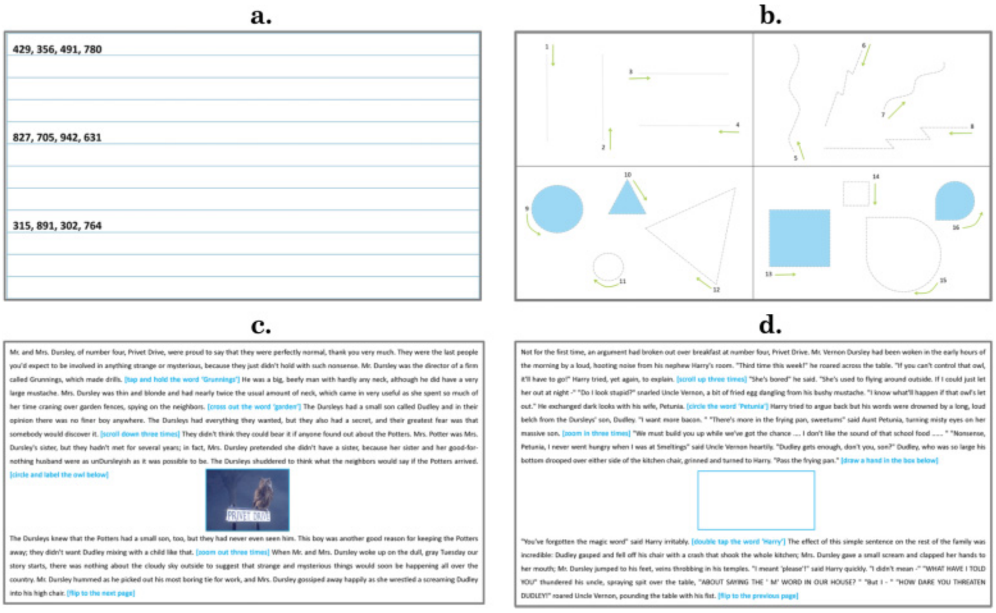
4.3. Tasks and Procedure

The experimental tasks were designed to capture realistic and natural inking behaviors. Each task was designed to encourage interaction in all areas of the screen, enabling digitizer data and behavior to be gathered from a variety of screen locations, not only the top or center of the screen. Three real-world inking tasks (i.e., writing, tracing, and document annotation) were designed for use in the study. As explained in the following text, the annotation task was split into two sections, so in reality, participants completed four tasks. To prevent fatigue and learning effects, the presentation order of the activities was counterbalanced across participants.

Participants were instructed to perform each task at a speed comfortable to them and no time limits were imposed. Participants were instructed to interact naturally, as if using pen and paper. To facilitate natural behavior, they were informed that touch input was disabled so that they could rest their hand, fingers, wrist, forearm, knuckles, and the like on the screen if needed. They were also reassured that such behavior would not produce any stray markings or accidental gestures and that they would not see any feedback about any intentional gestures they were asked to make (e.g., swipes). Although turning off touch input could influence the gestures that participants made, we deemed it necessary to prevent the digital-only hand postures that were observed previously by Annett et al. [2014b].

To focus the scope of the study, it was necessary to limit the number and type of tasks performed. Although activities such as web browsing with the stylus, dragging or repositioning content with the stylus, or using the stylus to tap on keyboard keys could have been examined, we wanted to begin the examination of unintended touch by focusing on the most common, and sought after, use case of styli, namely inking. As the chosen inking tasks required users to interact with all areas of the screen, the results should be somewhat generalizable to other scenarios as well. Future work could, of course, extend on our exploration to identify and understand novel hand postures or usage patterns that are unique in these scenarios and explore how they would implicate solutions to unintended touch.

4.3.1. Writing. In the writing task, participants were presented with three 12-digit numbers and were instructed to write out each number in words (e.g., 123,456,789,101 had to be written as "one hundred twenty-three billion, four hundred. . .; Figure 5(a)). Participants were instructed to write on the ruled lines similar to traditional ruled loose-leaf paper and were encouraged to use whichever spacing and writing style they wished (e.g., printing, handwriting, mixed). No touch-based interaction was needed for the writing task, leading all touch events captured by the digitizer to be classified as unintended. The writing task took between 5 and 10 minutes to complete.



Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense. Mr. Dursley was the director of a firm called Grunnings, which made drills. [tap and hold the word 'Grunnings'] He was a big, beefy man with hardly any neck, although he did have a very large mustache. Mrs. Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbors. [cross out the word 'garden'] The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere. The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. [scroll down three times] They didn't think they could bear it if anyone found out about the Potters. Mrs. Potter was Mrs. Dursley's sister, but they hadn't met for several years; in fact, Mrs. Dursley pretended she didn't have a sister, because her sister and her good-for-nothing husband were as unDursleyish as it was possible to be. The Dursleys shuddered to think what the neighbors would say if the Potters arrived. [circle and label the owl below]

The Dursleys knew that the Potters had a small son, too, but they had never even seen him. This boy was another good reason for keeping the Potters away; they didn't want Dudley mixing with a child like that. [zoom out three times] When Mr. and Mrs. Dursley woke up on the dull, gray Tuesday our story starts, there was nothing about the cloudy sky outside to suggest that strange and mysterious things would soon be happening all over the country. Mr. Dursley hummed as he picked out his most boring tie for work, and Mrs. Dursley gossiped away happily as she wrestled a screaming Dudley into his high chair. [flip to the next page]

Not for the first time, an argument had broken out over breakfast at number four, Privet Drive. Mr. Vernon Dursley had been woken in the early hours of the morning by a loud, hooting noise from his nephew Harry's room. "Third time this week!" he roared across the table. "If you can't control that owl, it'll have to go!" Harry tried, yet again, to explain. [scroll up three times] "She's bored" he said. "She's used to flying around outside. If I could just let her out at night -" "Do I look stupid?" snarled Uncle Vernon, a bit of fried egg dangling from his bushy mustache. "I know what'll happen if that owl's let out." He exchanged dark looks with his wife, Petunia. [circle the word 'Petunia'] Harry tried to argue back but his words were drowned by a long, loud belch from the Dursleys' son, Dudley. "I want more bacon. " "There's more in the frying pan, sweetums" said Aunt Petunia, turning misty eyes on her massive son. [zoom in three times] "We must build you up while we've got the chance .... I don't like the sound of that school food ..... " "Nonsense, Petunia, I never went hungry when I was at Smeltings" said Uncle Vernon heartily. "Dudley gets enough, don't you, son?" Dudley, who was so large his bottom drooped over either side of the kitchen chair, grinned and turned to Harry. "Pass the frying pan." [draw a hand in the box below]

"You've forgotten the magic word" said Harry irritably. [double tap the word 'Harry'] The effect of this simple sentence on the rest of the family was incredible: Dudley gasped and fell off his chair with a crash that shook the whole kitchen; Mrs. Dursley gave a small scream and clapped her hands to her mouth; Mr. Dursley jumped to his feet, veins throbbing in his temples. "I meant 'please'!" said Harry quickly. "I didn't mean -" "WHAT HAVE I TOLD YOU" thundered his uncle, spraying spit over the table, "ABOUT SAYING THE ' M' WORD IN OUR HOUSE? " "But I - " "HOW DARE YOU THREATEN DUDLEY!" roared Uncle Vernon, pounding the table with his fist. [flip to the previous page]

Fig. 5. The tasks used in the data collection experiment, (a) Writing (b), Tracing, (c) the first part of the Annotation task, and (d) the second part of the Annotation task.

4.3.2. Tracing. During the tracing task, participants were presented with a number of dashed lines and shapes (Figure 5(b)). Participants were instructed to trace along each line or shape in the direction of the arrow, in order, starting with the vertical line marked "1" and ending with the teardrop marked "'16." Whenever participants encountered a shaded shape, they were required to shade it in before continuing. A tracing task was chosen in place of a free-form drawing or sketching task to ensure that a variety of strokes and movements were performed by the participants and the movements would be relatively constant across participants. Similar to the writing task, no touch-based interaction was required. The tracing task took approximately 5 minutes to complete.

4.3.3. Annotation. In the annotation task, participants were presented with a short excerpt from a novel (Figures 5(c) and 5(d)). Embedded within the excerpt were short instructions that participants were required to perform. These instructions closely mimicked the behavior and actions found while reading or annotating a document, including "scroll down," "zoom in," "flip to the next page," "cross out a word," "circle and label," "tap and hold," "double tap," and "draw a hand." To prevent participants from performing each action in close succession to the previous one, and as many actions have a symmetrical counterpart (e.g., zoom in and out), the annotation task was divided into two parts. Participants thus performed the annotation task twice, using a different excerpt and set of target behaviors each time (i.e., one task contained "zoom in," "flip to the next page," "scroll up," and so on, and the other task contained "zoom out," "flip to the previous page," "'scroll down," and so on).

As touch was disabled across all tasks, no gesture recognition was implemented, nor did participants receive feedback once they made their gestures. Each annotation task lasted between 5 and 10 minutes. The embedding of such actions within the context of a larger reading task captured behavior in as close to a real-world situation as possible and ensured that the dataset to contain opportunities for simultaneous pen and touch interaction.

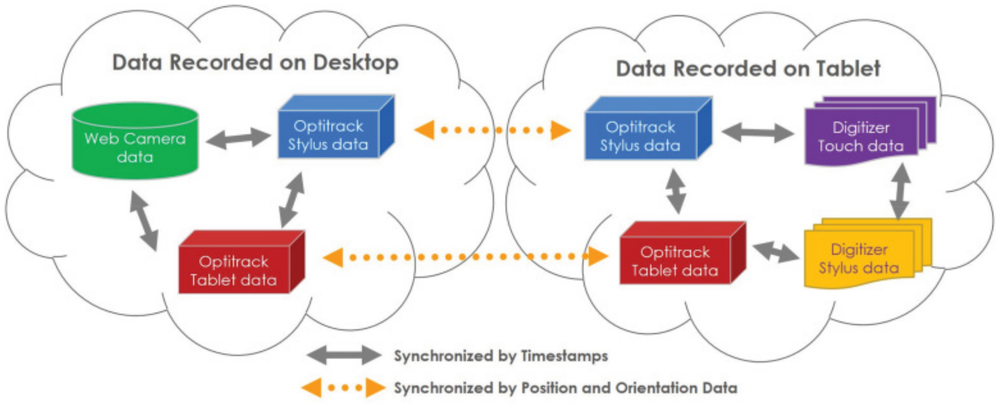


Fig. 6. Schematic of the data sources recorded and the synchronization performed.

4.4. Data Synchronization and Processing

The web cameras and Optitrack data were recorded on a desktop computer, and the stylus and touch data were recorded on the Sony tablet, thus data synchronization was necessary (Figure 6). The web camera and Optitrack data were synchronized using timestamps from the Windows clock, as they were on the same machine. The touch and stylus data were synchronized using a common OS-level clock for recording timestamps. As identical motion capture data was streamed to both the tablet and desktop and identical C++ clients were used to record the data, the streaming data (not the timestamps) was used to synchronize the data across both machines.

MATLAB was used to filter, process, resample, align, and synchronize the different data streams. Behavioral parameters such as the stylus height during inking and gesturing as well as the location and size of each touch event were also computed using MATLAB. The analysis and the comparison of unintended touch algorithms were performed using C++. When necessary, the VLC media application was used to analyze the video streams frame by frame.

5. EVALUATION OF SOLUTIONS TO UNINTENDED TOUCH

There are many factors and natural behaviors to take into consideration when designing approaches to unintended touch. Although several tablets report hover, pressure, tilt, and other sensor data, many do not. Approaches that make use of different sensor data and other information sources, such as hand models or user handedness, were thus analyzed.

5.1. Procedure

The ofxtouch OpenFrameworks [2009] connected-components algorithm was first applied to each frame of the collected data to group the activated sensors into "touch blobs." Once the data had been segmented, each blob was tracked frame to frame and assigned an ID using the ofxtouch blob tracking algorithm. In keeping with the desire to examine algorithms that introduced the smallest latency possible into the pipeline, the touch data from the first frame of the touch was used. This ensured that a decision would be made for every touch event, including those that lasted for only one frame. It also ensured that the speed of movement after the initial touch activation was not an influencing factor.

Although the location of the stylus was known for the duration of each task, for every frame of data collected, the motion capture data was used to determine whether the

Table I. The Confusion Matrix Used for the Unintended Touch Classification

	Intended Touch	Unintended Touch
Algorithm Accepts Touch	True Positive (TP) or Hit	False Positive (FP) or False Alarm
Algorithm Rejects Touch	False Negative (FN) or Miss	True Negative (TN) or Correct Rejection

stylus was located on, above, or beside the screen. Whenever the stylus was not on or directly above the screen, it was treated as being absent. This was a necessary because algorithms that required the presence of the stylus or the precise stylus location would be provided with incorrect data (given the antenna-based techniques used in current stylus-based systems to detect the stylus). Given that four of participants were left-handed, handedness-specific information was also integrated into the dataset to allow an analysis of those approaches that use spatially based information (e.g., ignoring everything to the right of the stylus for right-handers and to the left for left-handers). Although displays today do not detect handedness, this was deemed necessary for an unbiased analysis.

To assess the different unintended touch solutions, each touch was first classified as either intended or unintended. For the writing and tracing tasks, every touch event was classified as unintended, since no touch interaction was required during these tasks. In other words, 100% of the 5,085 touch events generated while writing and tracing were classified as unintentional and 0% were classified as intentional. For the annotation tasks, the only intentional touch events were those generated by the fingers while the participant was zooming, scrolling, flipping, and so on (see Section 4.3.3). To determine which touch events detected by the digitizer were intended or unintended, the web camera videos were consulted. As the order in which each participant performed each gesture was known a priori, the videos were manually analyzed to determine the timestamps corresponding to the beginning and ending of each gesture (i.e., "the intentional touch time period"). These timestamps were then used to classify each touch event as *intended* (i.e., the input detected by the digitizer occurred during an intentional touch time period and the videos illustrated that it was generated by one or more fingers) or *unintended* (i.e., the input detected by the digitizer occurred outside the intentional touch time period *or* the input detected by the digitizer occurred during an intentional touch time period but the videos demonstrated that it was not generated the fingers). Of the 1,218 touch events that were generated during the annotation task, 17% (i.e., 213) were classified as intentional and 83% (i.e., 1,005) were classified as unintentional. Although the total number and proportion of touch events generated during the writing/tracing and annotation tasks are not equal, they are representative of the disproportionate nature of intentional versus unintentional touch while inking.

After the intentionality of each touch event was determined, a contingency table (i.e., confusion matrix) was computed for each task, participant, and algorithm (Table I). In addition to being classified as intended or unintended, each touch event was further classified as either accepted or rejected. The acceptance or rejection of a given touch input was dependent on the parameters and techniques specified by the algorithm under assessment.

To determine the performance of various approaches to unintended touch, all of the cells in the contingency table were used to compute the *accuracy* of each algorithm (Equation (1)). A measure such as accuracy allows equal focus to be directed toward situations where the user intentionally interacted and the algorithm accepted the touch event (i.e., true positives) and situations where the user unintentionally interacted and the algorithm rejected the touch event (i.e., true negatives). Although one may assume that the only important aspect of rejection should be the true positives, due to their necessity for bimanual interaction, it is equally, if not more, important to consider how algorithms perform during tasks that do not require bimanual interaction, such as writing or drawing (i.e., evaluate the true negatives). Accuracy allowed such a comparison.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} * 100 \qquad (1)$$

While one could use measures such as precision and recall, such measures discount the importance of the true negatives, that is, unintended touch events that were correctly rejected. Given that users are more vocal about the stray inking and accidental navigation that is the result of poor rejection than their ability to perform gestures, it did not seem reasonable to use such measures. Different weightings could have been assigned to various cells of the contingency table, but we felt that this would have resulted in a biased analysis given the tasks users performed. Given the latency requirements detailed in Section 2, it would have also been appropriate to assess the running time of each algorithm. We opted not to perform this calculation and instead focus on a measure that was hardware invariant and allowed the algorithms to be implemented at anywhere along the data pipeline.

5.2. Algorithmic Approaches

There are many possible approaches to unintended touch. In this work, seven approaches, representative of the spectrum of possibilities today and the foreseeable near future, were evaluated. The approaches were chosen based on their prevalence in existing work, hypothesized efficiency, and dependence on additional data resources.

5.2.1. Reject All Touch Events. To set the context for our evaluation, the first approach that was evaluated was one in which all touch events, regardless of whether they were intended or unintended, were rejected. This is similar to the approaches taken by S Note and OneNote, where all touch input is rejected by the application. Although simple, such an approach effectively eliminates all the issues users reported with stray markings accidental navigation, while simultaneously preventing gesturing or any other desired touch-based input.

5.2.2. Contact Area. As access to the raw digitizer data was available, the *contact area* of each touch event was computed. The contact area was defined as the number of sensors activated by each individual touch event, on the first frame that the touch event was detected by the digitizer. This approach to unintended touch compared the contact area of each touch event to a number of different thresholds and rejected those touch events that had an area bigger than the threshold. For example, if a touch event covered an area of 5 sensors and the threshold was 10 sensors, the event would be accepted; if a touch event covered an area of 16 sensors and the threshold was 10 sensors, the event would be rejected.

As a number of different thresholds are possible, the thresholds that were evaluated in the present study were in the range of zero to 2,048 sensors (i.e., the maximum number of sensors on the device), in one-sensor increments (e.g., 0, 1, 2, 3). Although some iPad applications make use of the diameter of a touch contact to make rejection decisions and it is often the first approach many think of when imagining how they would solve unintended touch, the use of raw sensor data for unintended touch is novel. Such an approach could easily be integrated within the firmware, operating system, or specific applications.

5.2.3. Hover-Based Rejection. This hover-based approach mimics the behavior of existing Windows 7 and 8 devices. With these devices, whenever the stylus is close enough

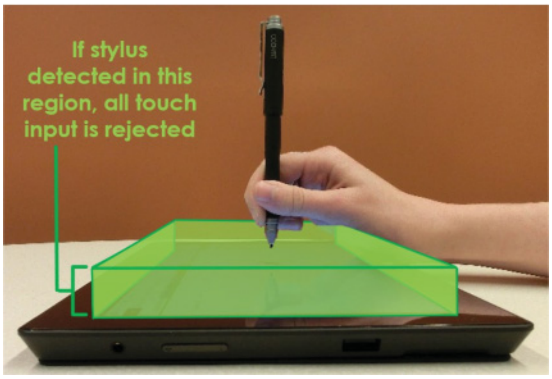

If stylus detected in this region, all touch input is rejected

Fig. 7. Example of the hover-based region wherein the all touch information would be rejected whenever the stylus is detected within the green region (box).

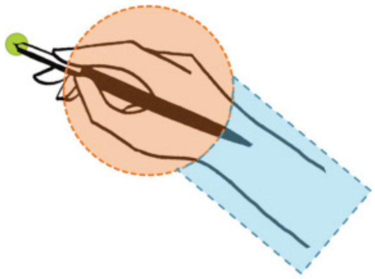


Fig. 8. Abstracted illustration of the Vogel Hand Occlusion Model, wherein the stylus location (the solid green circle) is used to dictate the location of the abstracted hand (i.e., the dashed orange circle) and the forearm (i.e., the dashed blue rectangle).

to the screen that it can be detected (i.e., the stylus is in the hover state; Figure 7), all touch input on the screen is ignored. For the present evaluation, the Optitrack data was used to compute the distance between the screen and the stylus's nib for each touch input. Using this information, whenever the height of the stylus's nib fell below a predetermined threshold, the whole touch input was rejected; if the nib height was above the threshold, the touch input was accepted.

Current devices detect the presence of the stylus up to a height of approximately 20mm, but with advances in sensor technology, this could increase further. Such an approach harnesses the hover state and could make hover-based approaches a viable option because they allow rejection decisions to start well before the skin touches the screen. It was thus of interest to evaluate this approach from a variety of possible detectable stylus heights (i.e., 0 to 200mm, in increments of 1mm).

5.2.4. Hand Occlusion Model. A generalized version of the Hand Occlusion Model algorithm by Vogel et al. [2009] was also implemented. In this model, the hand and wrist were abstracted into a circle and intersecting rectangle and were located at a specified distance and angle from the stylus (Figure 8). All touch events falling within the circle and rectangle are rejected, whereas those outside these areas are accepted. With this approach, if the stylus is not detected, then any touch input detected by the digitizer is accepted. As both right- and left-handed users participated in the data collection experiment, the angle and location of the "palm circle" and "forearm rectangle" were mirrored when the left-handed participants' data was used.

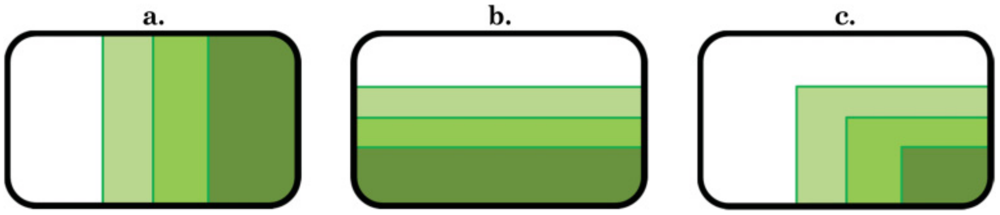


Fig. 9. The nine static rejection regions for right-handed users, with the area and location of the region being (a) vertical, (b) horizontal, or (c) intersecting horizontally and vertically. All touch events within the green "safe" areas would be rejected; those outside the green areas would be "unsafe and accepted." For left-handed users, the vertical regions would be mirrored horizontally.

The Hand Occlusion Model was evaluated because it has already been applied to overcome unintended touch in research [Yoon et al. 2013]. The model is hypothesized to work well for unintended touch because it maximizes the area available for (bimanual) interaction and ignores segments of the screen from where unintended touch is most likely (i.e., from the palm and forearm). In the present work, we decided to use the generalized model first proposed by Vogel et al. [2009]. Later work by Vogel and Balakrishnan [2010a] relied on the use of a calibration phase to generate personalized models for each user. However, given the additional calibration steps necessary, developer's resistance toward such steps, and the desire to ensure participants were naïve to the exploration of unintended touch (i.e., a calibration phase would have alerted users to the underlying goal of the exploration), the generalized model was used instead.

5.2.5. Static Rejection Regions. Inspired by the approach used with the Moleskine application, the use of "static" rejection regions was also evaluated. In these applications, rejection is performed by specifying a specific rectangular area of the screen that is "safe" to rest one's palm or forearm (Figure 9). An overlay is typically used to make the "safe" and "unsafe" regions of the screen visually distinguishable for users and remains persistent until the user decides to close them. Rejection that is performed using these regions looks at every touch input that is generated on the screen and compares the location of the touch input to the bounds of the "safe" region. If the input falls inside the region, the input is rejected; if the input falls outside the region, it is rejected.

Although Moleskin relegated their rejection region to cover the bottom third of the screen, there are many ways in which static rejection regions can be specified. Within the present investigation, three different region areas (i.e., filling 1/3 screen, 1/2 screen, and 2/3 screen; Figure 9) and three different spatial partitions (i.e., located vertically, horizontally, or having a horizontal–vertical intersection; Figure 9) were evaluated. The examination of the vertical and horizontal–vertical spatial partitions and the 1/2 screen and 2/3 screen areas are novel. The resulting nine rejection regions covered a range of possible areas, allowing for an extensive evaluation. As these approaches require only handedness information, rejection decisions can be made very quickly and along various segments of the data pipeline. For tablets that cannot distinguish between pen and touch or that do not support hover, static rejection regions could be a viable option.

5.2.6. Stylus-Based Rejection Regions. One of the downsides of the static rejection region approach is that the rejection region has little context or information about where one is currently interacting. As such, it cannot adapt to when one is inking in the top right corner (i.e., reject more of the screen) or in the bottom left corer (i.e., reject less of the screen and allow for intentional input almost everywhere else). A novel variant of the static rejection region approach that solves this issue is to use the current X, Y location

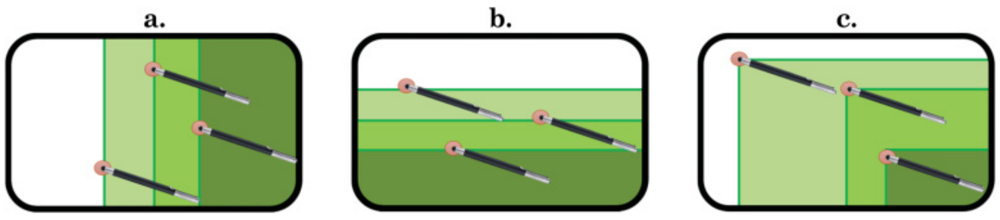


Fig. 10. Examples of stylus-based rejection regions for right-handed users, with the spatial partioning of the screen being (a) vertical, (b) horizontal, or (c) intersecting horizontally and vertically. The location of the region is dictated by the current stylus location (denoted by a solid pink circle). All touch events within the green "safe" areas would be rejected. For left-handed users, the vertical regions would be mirrored horizontally.

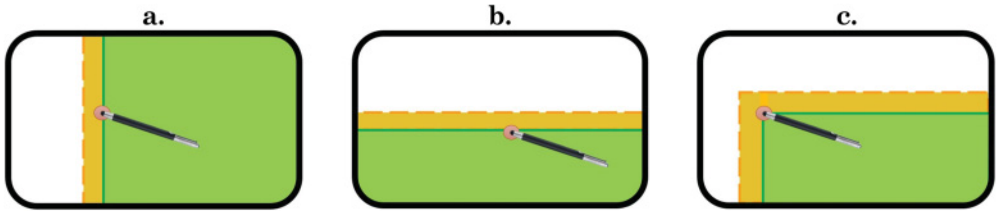


Fig. 11. An example of stylus-based rejection with a buffer with the spatial partioning of the screen being (a) vertical, (b) horizontal, or (c) intersecting horizontally and vertically. In this approach, the stylus-specified region extends away from the stylus location toward the dashed orange line. Touch events falling into the green "safe" region are rejected. For left-handed users, the vertical regions would be mirrored horizontally.

of the stylus to provide context for rejection. As the X,Y position indicates where one is currently interacting, it can dictate the location and bounds of a rectangular rejection region (Figure 10). Consider the use of a vertical rejection region. If the stylus is located at $x = 50$ and the participant is right-handed, a stylus-based rejection region would reject all touch events to the right of the stylus (i.e., $x > 50$) and accept those touch events that fall to the left of the stylus (i.e., $x < 50$). If the stylus were to move to $x = 600$, the rejection region would move as well, rejecting everything > 600 pixels and accepting everything < 600 pixels.

Similar to the static rejection regions, there are a number of spatial partitions that can be used to dictate the bounds of the rejection region. Within this exploration, stylus-based regions that were horizontal, vertical, or had a horizontal-vertical intersection were evaluated (Figure 10). Such regions should allow rejection to be performed quite quickly and along various segments of the data pipeline but does require the stylus location for the region to be specified.

5.2.7. Stylus-Based Rejection Regions Using a Buffer. Although the stylus-based rejection regions harnesses contextual information, they assume that the stylus will always be located above the palm, wrist, or forearm. As Annett et al. [2014b] demonstrated in prior work, not all hand postures follow this form. For example, hand postures that are inverted or "hooked," result in a nib location below or to the right of the palm or fingers. Based on the variety of hand postures observed, a novel variant of the stylus-based rejection region approach was evaluated. With this approach, a virtual "buffer" was extended out and away from the stylus-specified region (Figure 11) to accommodate such hand postures. Now, any touch input located within the stylus-specified region *or* the buffer region would be rejected; touch input in any other areas of the screen would be accepted. If we consider the vertical screen partition, what would happen if a 75 pixel buffer (~10cm) was used (and the user was right-handed)? If the stylus was

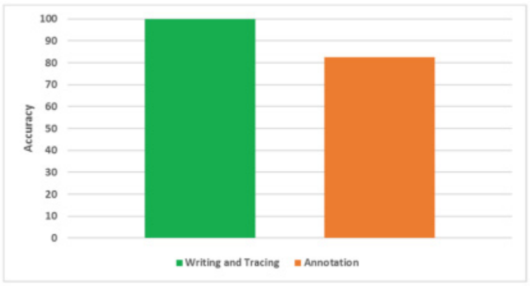


Fig. 12. The accuracy results for the Writing/Tracing and Annotation tasks when all touch events are rejected.

located at x = 875, for example, any touch input whose location was greater than x = 800 would be rejected, anything less than x = 800 would be accepted.

The use of "buffers" should accommodate the variety of hand postures used while inking [Annett et al. 2014b] and should be more robust to wrist and hand rotations [Fitzmaurice et al. 1999]. As the use of the stylus to specify the rejection region and "buffers" are novel, a variety of buffer dimensions ranging from 0 to 50mm were evaluated, in increments of 1mm. Similar to the static and stylus-based rejection regions, spatial partitions that were horizontal, vertical, or had a horizontal-vertical intersection were evaluated. This approach should allow rejection to be performed quite quickly and along various segments of the data pipeline, but of course requires that the stylus location is known.

5.3. Evaluation of Individual Algorithms

As the writing and tracing tasks did not contain any opportunities for intentional touch input, the proportion of intended versus unintended touch events within these segments of the dataset is skewed. Collapsing across the four tasks and combining all of the task data together would have resulted in a biased accuracy measure. Thus, we chose to separate the tasks into those that contained opportunities for touch input (i.e., the two annotation tasks) and those that did not (i.e., writing and tracing). By evaluating these datasets separately, we were able to determining how an approach would fair in situations that are increasingly likely in the future (i.e., bimanual input as represented by the annotation task) and those that users encounter today (i.e., writing and tracing without the desire for bimanual input). For the analysis, behavioral data gathered from the webcams, motion capture system, and digitizer were used where appropriate.

As there are many possible parameter variations applicable for a given approach (e.g., 200 different hover heights or 2,048 potential contact sizes) and each approach makes such of different data streams that were collected (e.g., hover height versus stylus location, others require both stylus and touch information), we first report on an individual analysis of each approach. Section 5.3.8 provides a summary of each algorithm's best results with respect to the raw contingency table values that were recorded. In Section 5.4, we identify the optimal parameters for each approach and utilize such information to statistically compare the algorithms against each other.

5.3.1. Reject All Touch Input. Unsurprisingly, taking an approach that rejects all touch input performs perfectly when no intentional touch input is generated, that is, with 100% accuracy during the Writing and Tracing tasks (Figure 12). When the Annotation task is considered, the accuracy is proportional to the amount of unintentional touch

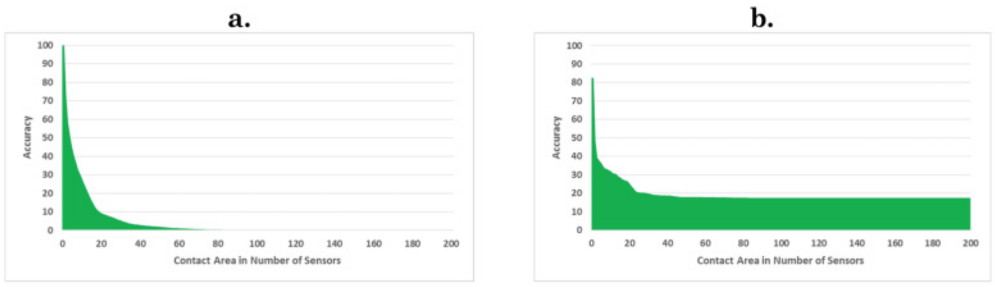


Fig. 13. Accuracy results for the (a) Writing/Tracing tasks and (b) Annotation tasks. There were not changes in accuracy from 200 to 2,048 sensors, so these values have been removed from the graphs for clarity.

input found in the dataset, that is, 83%. These results provide context and a baseline against which the remaining six approaches and their variations are considered.

5.3.2. Contact Area. The analysis determined that, regardless of opportunities afforded for intended touch, as the threshold of allowable contact areas increased, accuracy decreased (Figures 13(a) and 13(b)). Because the intersensor spacing of the digitizer was approximately 4mm, the average finger activated a median of approximately four sensors (IQR = three sensors). Rejecting touch events at this threshold resulted in 51% accuracy while writing/tracing and 38% accuracy while annotating. Once the allowable contact area increased beyond 4 sensors, accuracy continued to decrease until it leveled off at approximately 60 sensors (i.e., 1% while writing/tracing and 18% accuracy while annotating). The disparity found when annotating versus writing and tracing emphasizes the small contact area that most touch events, not just the fingers, activate when they initially touch the screen.

An analysis of the digitizer data makes it clear why the contact area approach is not sufficient. When we removed the intentional "finger" touch events from the dataset, that is, everything activating fewer than four sensors, and examining the initial area and growth of the remaining unintentional touch events, we found that most unintended touch events initially resembled a finger or group of fingers (Figure 14). If one looked exclusively at their size, intentional and unintentional touch events were thus indistinguishable. When we looked at the initial shape of the intentional versus unintentional touch events, they appeared visually indistinguishable. The activated sensors did not form ovals or circles, as one would expect [Murugappana et al. 2012]. The activated sensors often appeared as straight lines or irregular, jagged shapes (Figure 14, at 0ms). Given the indistinguishable nature of these events, the use of contact shape or size thus appears to be infeasible, as it is impossible to initially identify intentional finger-based touch input from other unintentional input.

We also looked at the growth of touch events over time to determine at which point an intentional touch event grows to be distinguishable from an unintentional touch event. It was determined that touch events initially activated a very small number of sensors (Mdn = six sensors, IQR = nine sensors) and did not grow to a differentiable size (i.e., larger than four sensors) until approximately 33ms after their initial activation (Figure 15). Such a slow rate of growth explains why a contact-based approach cannot be successful: both intended and unintended touch events initially appear the same to the digitizer and take much too long to become differentiable. Waiting for the contact areas to merge and stabilize before making a rejection decision is not feasible, especially when current system latencies are easily perceptible by users and less than 5ms are available for rejection decisions. Although the system recorded touch and stylus data from the digitizer at 60Hz, increasing the sensor density or sampling rate would have

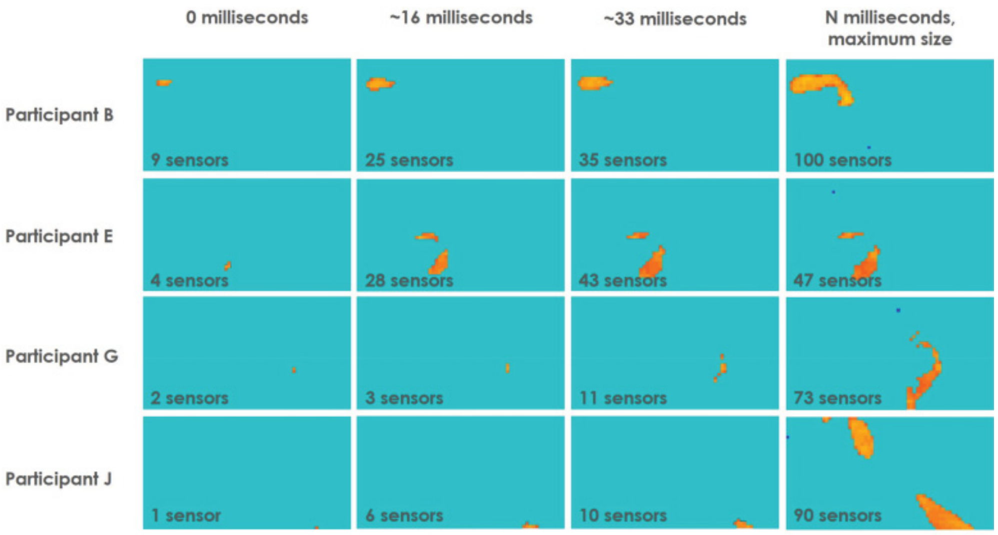


Fig. 14. Examples of different touch events that initially activated a small number of sensors (denoted by shades of orange) and later grew to their full size. Note that the initial shape and direction of the touch input does not represent the eventual shape and orientation of the touch input.

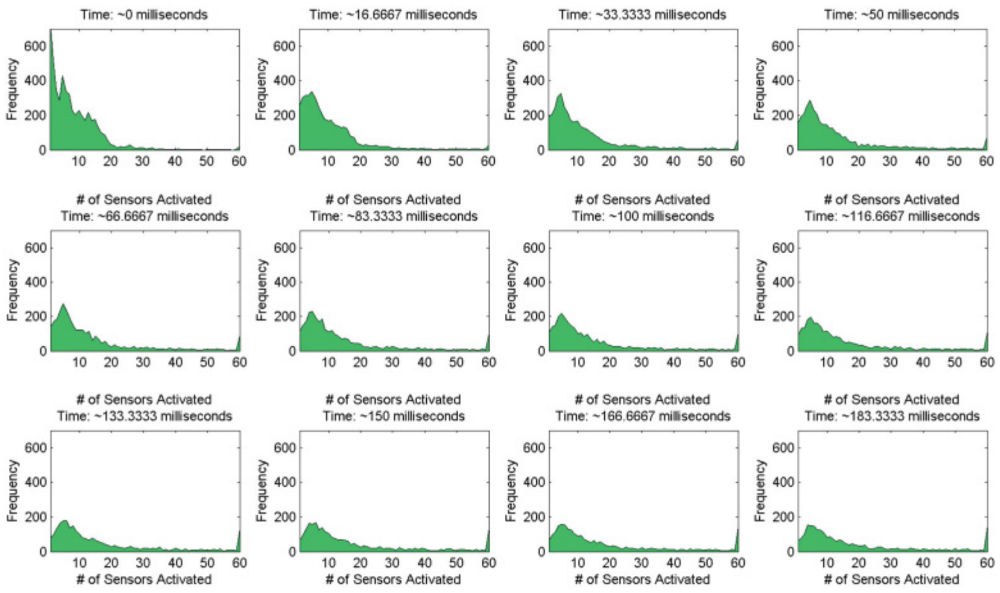


Fig. 15. The area of each non–finger touch event on the first, second, third, and so on frame of its existence.

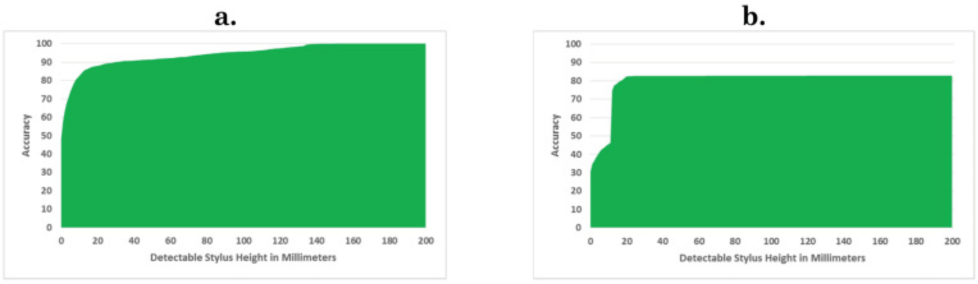


Fig. 16. Accuracy results for the (a) Writing/Tracing tasks and (b) Annotation tasks while the current height of the stylus was used for rejection.

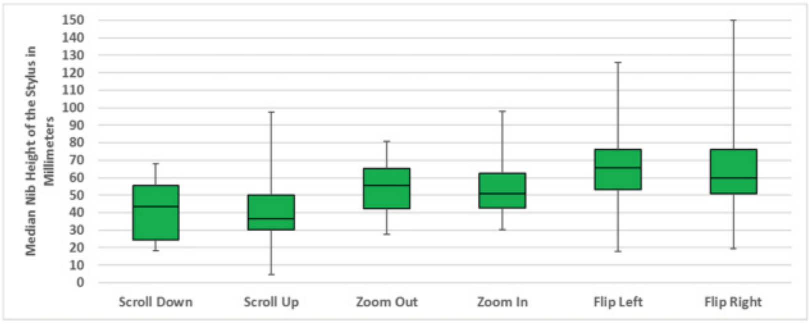


Fig. 17. The median nib heights measured during the different gestures performed throughout the Annotation tasks. Error bars represent the standard error of the mean. The measurement error of the motion capture system was within ±1mm.

no effect on the real-world speed at which the contact area grows, leaving the accuracy largely unchanged.

5.3.3. Hover-Based Rejection. The stylus height obtained from the motion capture system revealed that accuracy increases as the detectable hover height increases (Figure 16). At the hover height specified for Windows machines today (i.e., 20mm), correct rejection occurs 88% of the time when intended touch is not generated (i.e., writing/ tracing tasks) and 82% of the time when it is generated (i.e., annotation task). Whenever the detectable hover height is less than 20mm, accuracy drops to 31% to 47% when there is intentional touch interaction, but only 48% to 87% when there is only unintended interaction (i.e., during writing and tracing). The disparity between the two scenarios is likely a by-product of the natural height at which the user holds the stylus while interacting with their nondominant hand (see later discussion). With advances in hover height detection, the accuracy of hover-based rejection would continue to increase, achieving approximately 96% accuracy at 100mm and 100% accuracy at 200mm with no intended interaction, and 82% accuracy at 100mm and 83% accuracy at 200mm when intended interaction is permitted. This is similar to what was found for the Reject All Touch Input approach examined earlier.

As it is likely that hover heights will increase beyond the 20mm standard possible today, behavioral data gathered from the motion capture system suggests that hover will interrupt intentional unimanual and bimanual touch interaction. Across all of the gestures participants performed, the median nib heights were below 20mm except for the Zoom Out and Zoom In gestures (27mm and 30mm, respectively; Figure 17). These nib heights largely reflect the range of stylus-manipulation techniques participants

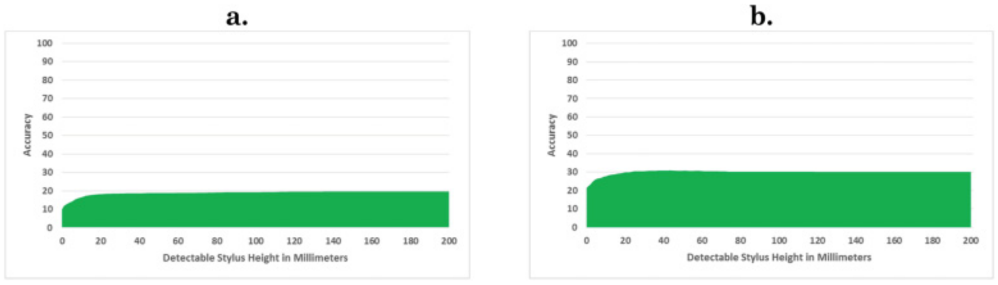


Fig. 18. Accuracy results while using the Hand Occlusion Model during rejection for the (a) Writing/Tracing tasks and (b) Annotation tasks.

used. Most participants tucked the stylus behind their index finger to allow the index finger and thumb to gesture (80%). Such gestures were often restricted to low, short flicking finger motions that ensured that the stylus nib stayed close to the screen. Other participants grasped or clenched all of their fingers around the stylus (10%) or used their nondominant hand to gesture (10%). The grasping motion allowed longer, more elongated gestures, which elevated the nib height, whereas when the nondominant was used to gesture, the stylus and hand did not change position. Although gesturing occurred infrequently during inking, the majority of these gestures would be rejected, aggravating users. Given these natural behaviors, it is important to consider the impact that raising hover heights above their current threshold levels would have on interaction. As the detectable hover heights increase, designers need to carefully consider how they inhibit or support various bimanual interactions.

5.3.4. Hand Occlusion Model. The Hand Occlusion Model, unfortunately, did not exhibit promising results (Figure 18). Across all detectable stylus heights, accuracy was slightly better during the annotation task. The hand posture parameters specified for the general model resulted in 18% accuracy (at 20mm) when no intentional interaction was generated and 30% accuracy when intentional interaction was permitted (i.e., during the annotation tasks). If stylus detection was possible at higher heights (i.e., up to 200mm), accuracy would marginally increase to 20% during writing and tracing, but stay at 30% while participants annotated. The disparity between the intentional and unintentional input scenarios (i.e., writing and tracing versus annotation) are likely due to the nature of the Hand Occlusion Model itself, which specifies the largest possible "safe" area for intended interaction. Although the model allows for increased intended interaction (as evidenced by Figure 18(b)), its lack of support for unintentional interaction resulted in the poor accuracy performance that was found.

The overall inadequacy of the Hand Occlusion Model is likely based on two factors, the frequency of stylus location information and the "shape" of the activated touch input. An analysis of the motion capture data determined that the majority of the generated touch events were rejected due to insufficient stylus data. As per the requirements of the model, whenever touch input was initiated and the stylus was not present, the touch input was accepted (i.e., treated as a false positive). The motion capture data revealed that participants held the stylus nib approximately 22mm (SEM = 9.27) above the surface when their palm first touched the screen. When transitioning the stylus from one line to the next, the stylus was held higher, with the nib approximately 47mm from the screen (SEM = 4.32). Such behaviors ensured that the stylus was outside the range currently detectable today and explain why more touch events were incorrectly accepted from 0 to 47mm. When we consulted the video data, we also found that many participants used the bottom or side device bezel to stabilize the arm (Figure 19). This

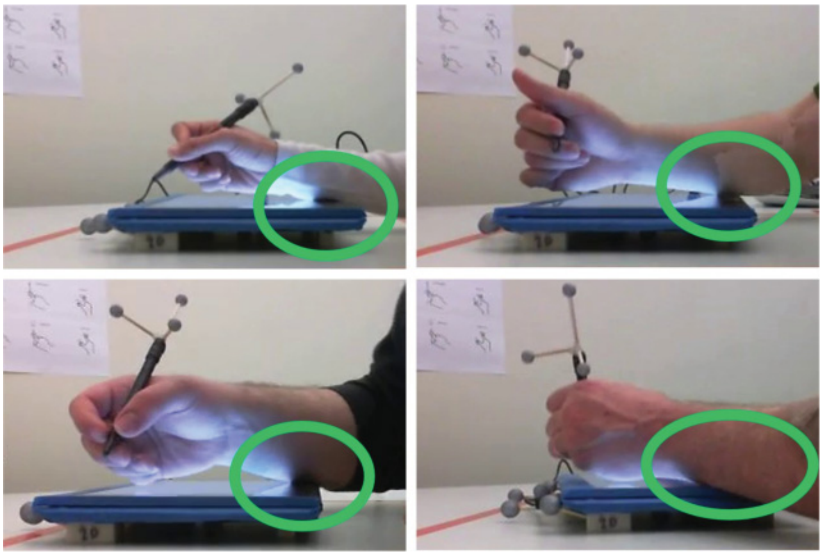


Fig. 19. Participants resting their forearm on the bottom bezel (and subsequently the bottom of the touch screen) that generated touch input (emphasized using a solid, green circle).

often resulted in the forearm crossing the bezel and generating touch input on the screen, much in advance of the wrist, palm, fingers, and stylus touching the screen. As the stylus height during this behavior far exceeded the detectable regions we tested, these touch events were incorrectly accepted, even when the stylus was detected at 200mm.

The generalized Hand Occlusion Model was also insufficient when one considers the shape of activation generated by the participant's hand postures. Some participants used an inverted or "hooked" hand posture [Levy and Reid 1978] (Figures 20(a), 20(c), 20(d)), whereas others used a more natural posture [Annett et al. 2014b], holding the stylus such that the nib was in a diagonal line with the palm and forearm (Figure 20(b)). These behaviors resulted in a majority of unintended touch input falling outside the generalized circle of the palm or rectangle of the forearm. While creating a personalized version of the hand model that could rotate farther from the nib (e.g., Figure 20(c)), translate closer to the nib (e.g., Figure 20(b)), or rotate the "forearm" rectangle (e.g., Figures 20(a) and 20(d)) could eliminate some of these issues, the model would need to be continually updated on a stroke-by-stroke [Fitzmaurice et al. 1999], task-by-task [Annett et al. 2014b], or device-by-device basis. This would degrade the user experience and likely introduce additional application-specific latency into the latency pipeline.

5.3.5. Static Rejection Regions. The static rejection region approaches performed quite well (Figure 21), partitioning the screen vertically (maximum rejection of 89% for writing/tracing and 78% for annotation) was superior to horizontally (maximum rejection of 69% for writing/tracing and 75% for annotation) and horizontally was superior to using a vertical-horizontal intersection (maximum rejection of 58% for writing/tracing and 44% for annotation). When considering the proportion of the screen "safe" for interaction during writing/tracing, those covering two thirds of the screen (maximum rejection of 89%) were superior to those covering half the screen (maximum rejection of 70%) and a third of the screen (maximum 49%). For the annotation tasks, those regions that covered two thirds of the screen (maximum rejection of 78%) were superior to those covering half the screen (maximum rejection of 72%) and a third

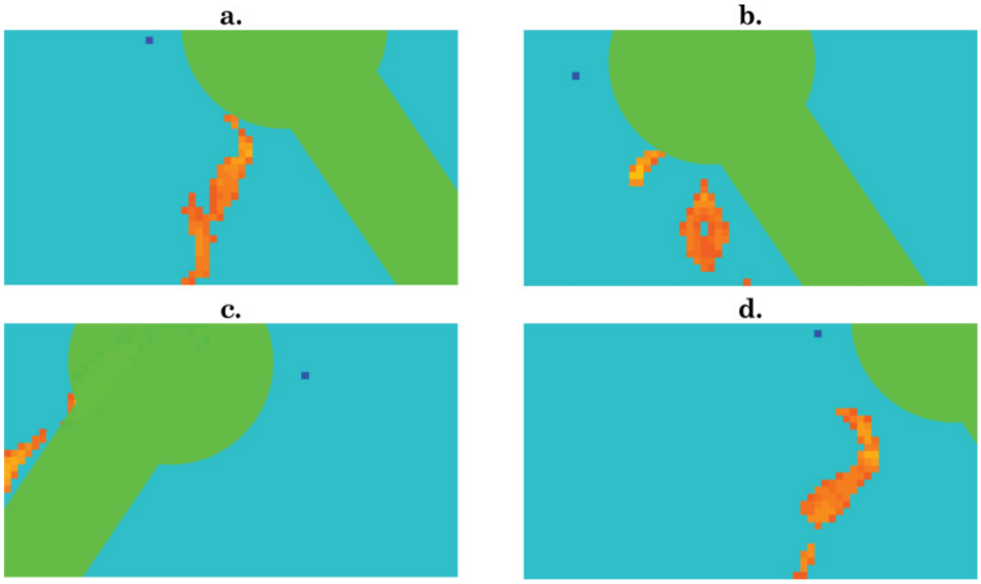


Fig. 20. Examples of various hand postures that fell outside the generalized Hand Occlusion Model. The skin input is denoted in shades of orange, the stylus location is denoted in blue, and the Hand Occlusion Model's abstracted circle and rectangle are green.

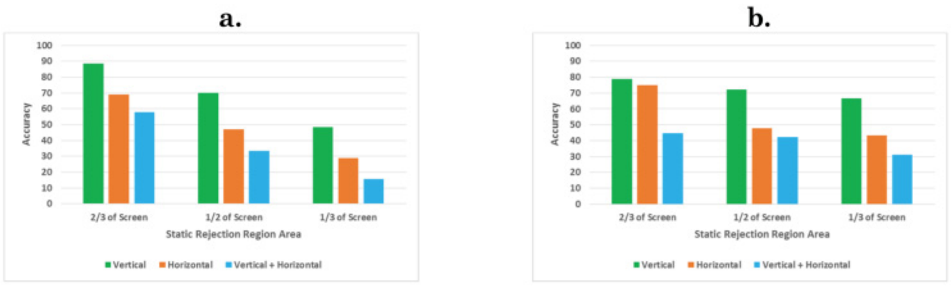


Fig. 21. The accuracy found for various static region-based approaches while participants were (a) Writing/Tracing and (b) Annotating.

of the screen (maximum 66%). Across all manipulations, a maximum of 87% correct rejection was found for the writing/tracing tasks and 79% for the annotation tasks. Thus, there appears to be a general trend toward screen partitions that rejected touch input from the largest vertical area. Vertical rejection regions likely work well because they capitalize on the natural ergonomics and kinetics of the lower arm. When on the screen, the wrist, forearm, and hand generally fall along a diagonal or vertical line and are normally oriented vertical instead of horizontal.

To better understand the performance of the various configurations, the locations of all touch inputs generated by participants were graphed using heat maps (Figure 22). In the writing tasks, most touch events occurred in the bottom right quadrant of the screen, where the forearm, wrist, and palm rested while right-handed participants were writing (Figure 22(a)). Surprisingly, touch input was not simply mirrored to the bottom left for the left-handed participants. Instead, left-handed participants exhibited unique behaviors, generating many more touch events along the left and top edges. In the tracing tasks, touch input for both handedness populations was much more dispersed

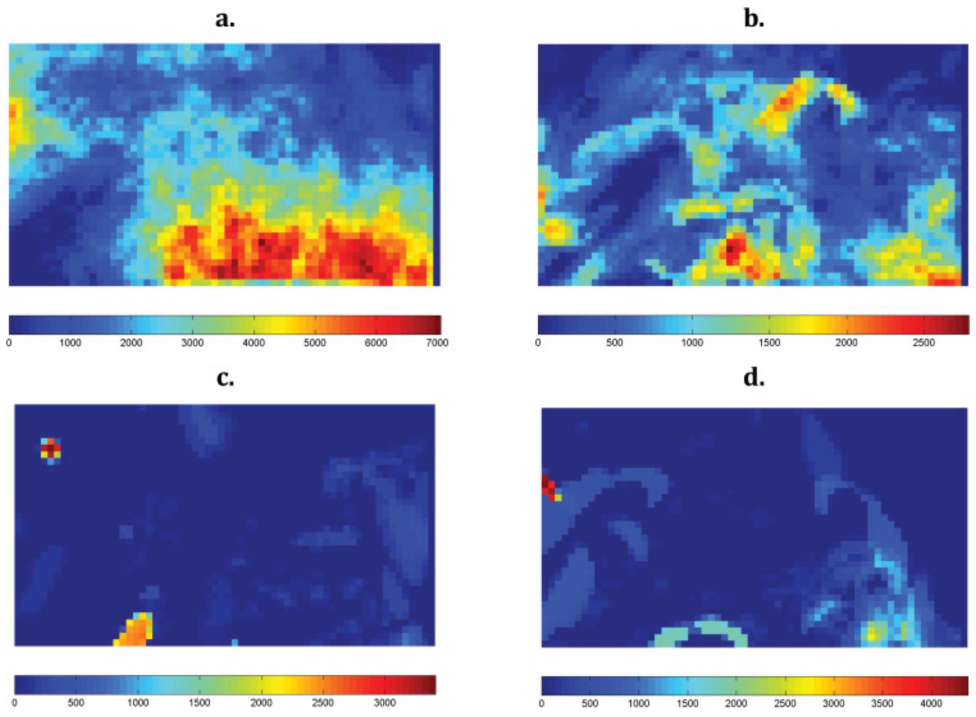


Fig. 22. Heat maps of all touch input generated by all participants for the (a) Writing, (b) Tracing, (c) the first Annotation task, and (d) the second Annotation task. The heat map data has not been normalized.

and found in many different locations of the screen (Figure 22(b)). The hot spots found in the center and along the left and bottom right bezels are indicative of areas where users rested their arms on the screen while completing the drawing tasks in each quadrant. As much less interaction was required in the annotation tasks, it is not surprising that input was found in the same areas for all participants (and somewhat mirrored for left-handed participants, Figures 22(c) and 22(d)). As interaction was restricted to key areas for each participant, touch input patterns were concentrated in these locations.

These task-dependent differences help explain why static rejection regions are a popular and reasonable first choice for unintended touch. They harness the task context and hypothetical behavior found with these tasks to eliminate the majority of unintended touch events (e.g., those occurring in the bottom right for right-handed participants). As writing and tracing required interaction in all areas of the screen, it is unsurprising that the larger rejection regions worked better than the smaller ones. Although the typical distance between the hands, while performing bimanual actions, has yet to be explored, the similar results found between the writing/tracing and annotation accuracy results suggest that the 2/3 vertical split left enough room for users to interact bimanually when they needed to (i.e., during the annotation task).

5.3.6. Stylus-Based Rejection Regions. The use of the stylus to specify the location of various rejection regions showed a similar trend to the static rejection regions: the use of a vertical region was better than a horizontal region, and both were better than a horizontal-vertical intersection (Figure 23). Based on the distribution of touch events presented in Figure 22, it makes sense that approaches that were most accommodating to unintended touch and the natural behavior or users would be successful. Across both writing/tracing and annotation, increasing the detectable stylus height (i.e., the

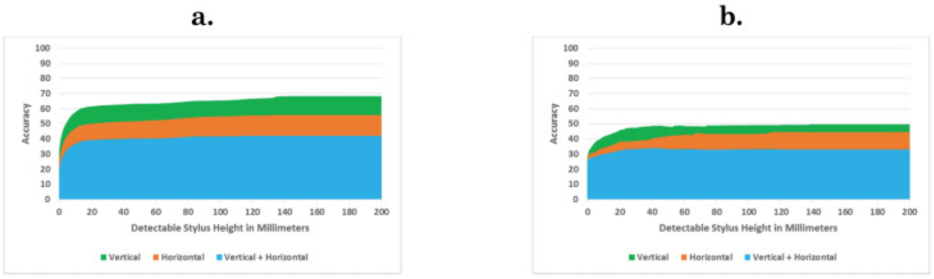


Fig. 23. The accuracy of stylus-based rejection while using different rejection regions and when the stylus is detectable from different heights. Image (a) contains data from the Writing/Tracing tasks and (b) contains data gathered during the Annotation tasks.

instances where touch events had corresponding stylus information), lead to a slight increase in performance. Given the technical limitations of stylus detection today (i.e., up to 20mm), a maximum of 62% accuracy was possible with the vertical partitioning for the writing/tracing tasks and 46% for the annotation task. If a tablet could detect the stylus up to 200mm from the screen, accuracy increased to 68% and 50%, respectively.

What was found to differ with stylus-based regions is that accuracy decreased when intentional touch was supported (i.e., annotation tasks). The nature of the touch and stylus input was quite different during the annotation tasks. Instead of being able to drag the hand around the screen from left to right and top to bottom, the annotation task naturally encouraged replanting behavior [Annett et al. 2014b], that is, participants placed their hand on the screen, performed an action, picked it up and held it in the air, replanted it in a different area of the screen, and so on. This intermittent behavior resulted in many more touch events that could not be classified because information about the stylus was missing. Similar to the Hand Occlusion Model, the success of stylus-based rejection regions depends on the availability of the stylus location. Although the location of the stylus was tracked at all times, the higher the stylus is in the air, the more uncertainty there is regarding where it would eventually land on the screen due to the range of motion possible via the degrees of freedom afforded by the elbow, wrist, and fingers. This uncertainty manifested itself in decreased accuracy.

The decreased accuracy across all tasks exhibited by the stylus-based rejection regions (compared to the hover and static rejection regions) is also due to the variability of hand postures that could be used while interacting. Although information about the stylus location could provide context as to where the user is interacting, the video and touch data revealed that such context information is not very useful. An analysis of the hand and wrist postures determined that the use of natural and inverted hand postures placed contacts outside the specified rejection area (Figure 24). On many occasions, the stylus location fell in line (Figure 24(b)), far below (Figures 24(a) and 24(d)–(f)), or to the side (Figure 24(c)) of the touch input in question. For this reason, the largest rejection region, the vertical screen partition, was the most tolerant to nontraditional hand postures and rotations of the wrist. Ignoring only the touch events in the bottom or bottom right corner was too inflexible, as the hand reorients as it moves toward the bezels and bottom of the screen. The use of larger "safe" areas accommodates the potential movement of the wrist and hand throughout the task.

5.3.7. Stylus-Based Rejection Regions Using a Buffer. The results obtained with the stylus-based rejections using a buffer demonstrate the same general trend as the static rejection regions and stylus-based rejection regions: the use of a vertical partioning worked best for rejection (i.e., maximum accuracy of 80% accuracy while writing/tracing, and

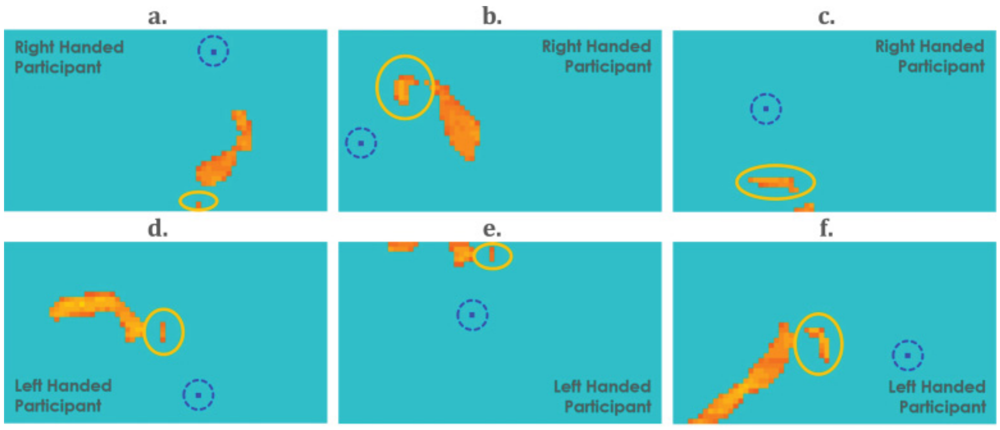


Fig. 24. Examples of various touch inputs generated by participants that would be incorrectly accepted when the stylus location is used to define the rejection region. The skin contact is denoted in shades of orange, the touch input under consideration is circled with a solid orange line, and the stylus location is highlighted using a blue dashed line.

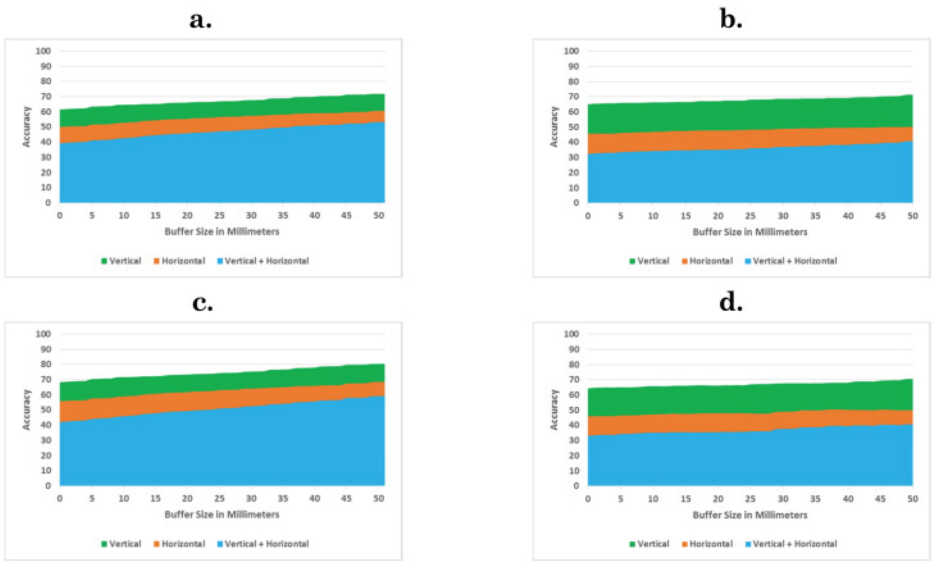


Fig. 25. Accuracy of the stylus-based rejection region approaches with different rejection partitions while (a,c) Writing/Tracing and (b,d) Annotating. Images (a,b) illustrate the accuracy when the stylus is detectable from 0 to 20mm and (c,d) illustrate the accuracy when the stylus is detectable from 0 to 200mm.

71% accuracy while annotating; Figure 25). Given the distribution of touch events in Figure 22 and the vertical region being attuned to the anatomy of the hand, these results underscore the importance of harnessing natural user behavior and observations for unintended touch.

The results also illustrate that as the size of the buffer increased, so did the accuracy (e.g., the vertical partition saw an increase from 62% to 72%, Figure 25(a); from 65% to 72%, Figure 25(b); from 68% to 80%, Figure 25(c); from 64% to 71%, Figure 25(d)). When we consider this “best case,” that is, vertical partition, and the stylus is detectable from 0 to 20mm, a 50mm buffer will result in 72% accuracy when only unintentional

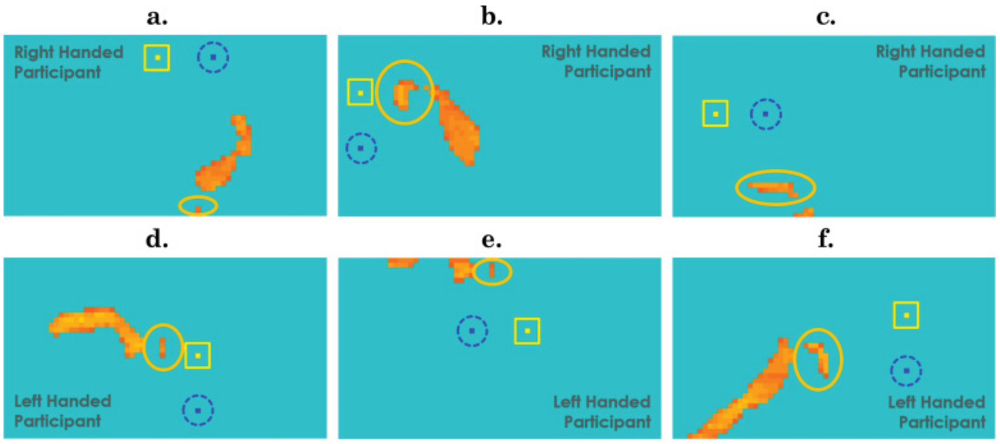


Fig. 26. A demonstration of how the use of a buffer can accommodate many of the unintended touch events that were incorrectly accepted in Figure 25. The skin input is denoted in shades of orange, the current touch blob under consideration is circled via a solid orange line, the detected stylus location is highlighted by a blue dashed circle, and the resulting stylus location after a 50mm buffer was added is highlighted via a yellow rectangle.

touch is considered (i.e.,, writing and tracing) and 71% accuracy when intended and unintended touch is allowed (i.e., annotation). If the stylus can be detected from 0 to 200mm, accuracy increases slightly to 80% while writing/tracing and stays the same while annotating. Although we did not statistically analyze the writing/tracing and annotation tasks, there was little difference between them, save the slightly higher variance between the spatial partitions during annotation. The results thus demonstrate that the larger the buffer region, the greater the increase in accuracy, illustrating that the buffer succeeded in accommodating the variety of hand postures likely during inking and that rejection was largely not influenced by the desire to perform intended versus unintended touch.

The touch-based data from Figure 22 demonstrate that participants may not always use a "natural" hand posture. It is thus not surprising that an approach that accommodates variations in posture, grip, or general anatomy is more successful than one that cannot. When we reassess the instances where user behavior resulted in the stylus-based rejection regions failing (from Figure 25), the use of a buffer would eliminate the unintended touch created by the palm wrist, fingers, and forearms (Figure 26). Although the buffer does increase the likelihood that hand posture can be accounted for, it does, of course, require that the stylus location is always known. Although the use of a buffer resulted in better performance, there were still a number of unintended touch events that were accepted due to the stylus being above or outside the detectable stylus range (and thus the rejection region location and boundaries could not be determined).

Given that the palm or forearm often touches the surface before the stylus, these results provide even more support for acquiring data as soon as possible. If the stylus location was always known, possibly using technology other than what is available today, this approach could work better because it can harnesses contextual information about where the user will be interacting. When combined with handedness, it becomes very powerful, allowing for a variety of hand postures and supporting many bimanual input techniques.

5.3.8. Aggregated Contingency Table. As many parameters and variations were evaluated for each algorithm, it would be unwieldy to present the raw contingency tables for each variation that was examined. Thus, an aggregated summary of the raw

Table II. A Summary of the Aggregated Raw Contingency Tables Found for Each Algorithm That Was Evaluated
The percentage of true positives (TP), true negatives (TN), false positive (FP), and false negatives (FN) are presented within the brackets.

Algorithm	Writing/Tracing				Annotation			
	TP	TN	FP	FN	TP	TN	FP	FN
No Algorithm	0	5,085 (100%)	0	0	0	1,005 (83%)	0	213 (17%)
Contact Area[1]	0	2,596 (51%)	2,489 (49%)	0	98 (8%)	362 (30%)	643 (53%)	115 (9%)
Hover[2]	0	4,474 (88%)	611 (12%)	0	204 (17%)	800 (66%)	205 (17%)	9 (1%)
Hand Occlusion Model[3]	0	926 (18%)	4,159 (82%)	0	212 (17%)	152 (12%)	853 (66%)	1 (0%)
Static Rejection Region[4]	0	4,506 (89%)	579 (11%)	0	42 (3%)	916 (75%)	89 (7%)	171 (14%)
Stylus-Based Rejection Region[5]	0	3,128 (62%)	1,957 (38%)	0	212 (17%)	345 (28%)	659 (54%)	2 (0%)
Stylus-Based Rejection Region with Buffer[6]	0	3,653 (72%)	1,432 (28%)	0	208 (17%)	403 (33%)	602 (49%)	5 (0%)

[1]Contact area of four sensors.
[2]A detectable stylus height of 20mm.
[3]A detectable stylus height of 20mm.
[4]A vertical rejection region covering 2/3 of the screen.
[5]A detectable stylus height of 20mm with a vertical rejection region.
[6]A detectable stylus height of 20mm with a vertical rejection region and a 50mm buffer.

contingency table values (i.e., number of the true positives, true negatives, false positives, and false negatives) can be found in Table II. The presented data illustrate the best results found for each algorithm given the technological capabilities of today (i.e., the detectability of the stylus up to 20mm) and the most reasonable parameters.

5.4. Comparative Evaluation of Algorithms

The exploration thus far has focused on algorithmic approaches that have many possible variations and options. Participants' behavior helped explain why some of the approaches were successful in natural inking settings, and found the same trends regardless of the task and underlying amount of intentional versus unintentional touch input. At the same time, the results underscored the importance of understanding user behavior during inking and the complexity of unintended touch. What is missing, however, is a comparison and understanding of the approaches that are best suited for unintended touch, equally weighing accuracy and the limitations of each approach, the required functionality, and how they would impact the cohesion of user experiences across the stylus and tablet ecosystem.

5.4.1. Methodology. As each of the approaches use different types of data and rely on different assumptions, an analysis was performed of how they fared against each other when the technological restrictions of tablets and styli were removed. A comparison of each approach, from the perspective of today's technology and the technological improvements likely in the foreseeable future, was then performed using the optimal parameters and results for each algorithm from Section 5.3 (Table III). Given that the same trends were found across the writing/tracing and annotation tasks, the data was aggregated to remove task as a factor. Note that even with technological advancements, not all approaches will improve (e.g., increases in sensor density will increase the number of sensors that are activated uniformly).

Table III. Parameters Used in the Comparison of Different Unintended Touch Algorithms Given Current and Future Hardware and Software Functionality

Algorithm	Present Parameters	Future Parameters
Contact Area	• 4 sensors	• 4 sensors
Hover	• 20mm hover height	• 200mm hover height
Hand Occlusion Model	• Stylus location known within 20mm of surface	• Stylus location known within 200mm of surface
Static Rejection Region	• Using vertical rejection region covering 2/3 of the screen	• Using vertical rejection region covering 2/3 of the screen
Stylus-Based Rejection Region	• Using vertical rejection region • Stylus location known within 20mm of surface	• Using vertical rejection region • Stylus location known within 200mm of surface
Stylus-Based Rejection Region with Buffer	• Using vertical rejection region • Stylus location known within 20mm of surface • Buffer of 50mm	• Using vertical rejection region • Stylus location known within 200mm of surface • Buffer of 50mm

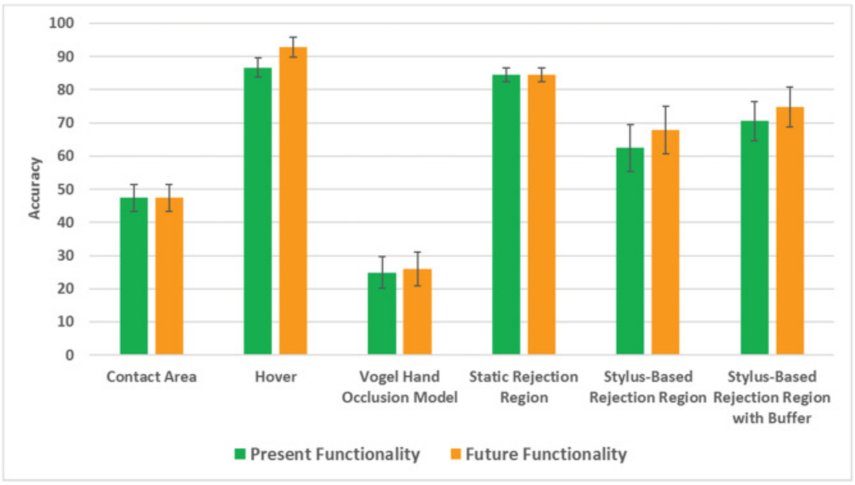


Fig. 27. Comparison of the unintended touch algorithms evaluated using present day and hypothetical future functionality.

Using the parameters in Table III, a two-way ANOVA with the factors of Functionality (Present, Future) and Algorithm (Contact Area, Hover, Hand Occlusion Model, Static Rejection Region, Stylus-Based Rejection Region, and Stylus-Based Rejection Region with a Buffer) was run using the accuracy values determined from the contingency table for each participant.

5.4.2. Results. The ANOVA results demonstrated that Functionality significantly influenced accuracy ($F_{1,15} = 5.0$, $p < .05$) as did Algorithm ($F_{2.3,35.0} = 30.6$, $p < .001$; Figure 27). There was no evidence of a Functionality by Algorithm interaction ($F_{1.4,20.8} = 2.6$, $p = 0.112$). Increases in technological functionality and capabilities led to small, but significant, enhancements in the performance of some algorithms that were evaluated (*Present*: M = 62.73, SEM = 3.00; *Future*: M = 65.56, SEM = 3.20; $p < .05$), namely the Hover (i.e., 6% improvement in accuracy), Stylus-Based Rejection Region (i.e., 5% increase in accuracy), and Stylus-Based Rejection Region with Buffer (i.e., 4% increase in accuracy) approaches.

Post-hoc Bonferroni-adjusted paired *t*-tests demonstrate that some approaches are much better suited for unintended touch than others (Table IV). Approaches that made use of stylus data or supported larger "safe" areas of the screen (i.e., Hover, Static

Table IV. Pairwise Comparisons between Unintended Touch Algorithms

Pairwise Comparison	ΔM	SEM	*p*
Contact Area vs. Hover	–42.3	5.5	0.015*
Contact Area vs. Hand Occlusion Model	21.9	5.3	0.015*
Contact Area vs. Static Rejection Region	–37.1	4.6	0.015*
Contact Area vs. Stylus-Based Rejection Region	–17.7	7.9	0.63
Contact Area vs. Stylus-Based Rejection Region with Buffer	–25.3	6.8	0.03*
Hover vs. Hand Occlusion Model	64.3	6.5	0.015*
Hover vs. Static Rejection Region	5.3	2.3	0.57
Hover vs. Stylus-Based Rejection Region	24.7	7.6	0.075
Hover vs. Stylus-Based Rejection Region with Buffer	17.1	5.8	0.15
Hand Occlusion Model vs. Static Rejection Region	–59.0	6.3	0.015*
Hand Occlusion Model vs. Stylus-Based Rejection Region	–39.6	6.6	0.015*
Hand Occlusion Model vs. Stylus-Based Rejection Region with Buffer	–47.2	6.4	0.015*
Static Rejection Region vs. Stylus-Based Rejection Region	19.4	8.2	0.48
Static Rejection Region vs. Stylus-Based Rejection Region with Buffer	11.8	6.6	1.44
Stylus Rejection Region vs. Stylus-Based Rejection Region with Buffer	–7.6	2.3	0.09

*denotes significance.

Rejection Region, Stylus-Based Rejection Region, and Stylus-Based Rejection Region with Buffer) outperformed those using generalized data sources with smaller "safe" areas, such as Contact Area or the Hand Occlusion Model. It thus appears that the size of the "safe" versus "unsafe" areas and the use of contextual and peripheral information to specify the location of these areas (e.g., current task, stylus height, stylus location) are important to unintended touch.

5.4.3. Discussion. The comparative analysis found a significant difference between Contact Area and the Hand Occlusion Model. Given that the Hand Occlusion Model relies on the stylus location and has a very narrow spatial area within which touch input is allowed, it is not surprising that the Hand Occlusion Model performed worst of all the approaches. As shown in the individual analysis, participants' hand postures were quite variable, rarely conforming to the abstract generalization that the Hand Occlusion Model required. Although one could make use of a trained, adaptive hand model, the task-by-task heat maps show that, depending on the requirements of a task, participants exhibit drastically different hand patterns from start to finish, especially in the more free-form tasks, such as tracing and annotating. Apart from the problem of variable hand behavior, personalized hand models would need to be relearned when transitioning to a new device, form factor, or task. This would likely involve a calibration phase, something many developers are very resistant towards.

By comparison to the other four approaches, the use of Contact Area performed no better than chance, due in large part to the length of time necessary for each touch event to grow to a distinguishable size before rejection could occur. For the Contact Area approach to be more successful, touch input would need to be provided before skin touches the screen (i.e., when the hand or forearm is within the "hover" state). This would then allow the disambiguation and rejection processes to begin before touch input activates the digitizer and possibly soon enough for correct rejection and acceptance to occur.

As there are no significant differences between the top four approaches, it cannot be claimed that one is superior to the others. There are a number of marginally significant results, however, that can be used to better understand where to devote research resources in the future. The Hover approach is significantly, or marginally significantly, better at unintended touch than the other approaches. In essence, the use of hover creates the largest rejection region: the whole area of the screen. This makes it the most tolerant to nontraditional hand movements and positions but limits its usefulness for scenarios where bimanual interaction is desired. As the rejection region gets smaller, either because it is bounded by the stylus location, the buffer dimensions, or the screen partitioning, the success of space-based approaches decreases. The results found with the Stylus-Based and Static Rejection Regions support this conclusion.

The marginal difference between the Buffer and Non-Buffer Stylus-Based Rejection Region approaches demonstrates that allowing a “spatial cushion” around the stylus is a simple but effective way to accommodate rotations of the wrist and palm. Furthermore, the relatively small rejection area allows for some bimanual interaction, which is not supported by the hover-only approach. Future approaches that use stylus location should consider integrating a similar spatial cushion. Based on the heat map distributions presented earlier, the Static Rejection Region approach did a good job of rejecting unintended touch in tasks where the hand is anchored on the screen in a predictable manner (i.e., while writing). As no stylus information is needed, rejection occurs quickly, and it does not result in false positives whenever the stylus location is not present. With applications such as note-taking on capacitive devices, it would be beneficial to use such a region, as it would prevent most unintended touch interaction. Coupling a visual representation with the rejection region allows users to understand where they can rest their palm without disrupting their work and increases the likelihood they will write as naturally as possible.

It is important to note that the three approaches that make use of stylus information are influenced by the detectability of the stylus. When the stylus location is not known (i.e., above 20mm or 200mm for the Present and Future conditions), the algorithms report false positives. Similar to the touch information required for the contact size, it would also be beneficial to receive stylus and touch input as soon as possible, before the stylus or skin touches the screen if it is technologically feasible.

6. OVERALL DISCUSSION

The present exploration indicates that there are many elements that contribute to unintended touch, making the problem, and choice of an unintended touch solution, very complex. Although our underlying dataset does not contain an equal number of intentional versus unintentional touch events, we believe that the dataset and results are representative of the distribution of such events common during real-world inking tasks, wherein the frequency of unintended touch almost always outweighs intended touch. For example, while taking notes during a lecture, it is likely users would only intentionally touch the screen whenever they needed to scroll the page. Such behavior results in a drastically smaller proportion of intentional compared to unintended touch events, due in large part to the frequency of the hand dragging or being replanted on the screen [Annett et al. 2014b] compared to page scrolling. As such, approaches that focus on, or are slightly biased toward identifying unintended touch instead of intended touch (e.g., Hover, Static Rejection Region) would perform better (e.g., Stylus-Based Rejection Region, Stylus-Based Rejection Region with a Buffer).

The similar trends found in the individual algorithm analysis, exemplifies that the more opportunities and support are provided for intentional touch, the more contextual knowledge regarding user behavior is required. If one considers a sketching application that allows users to zoom in and out via a pinching gesture to add details or perform shading, for example, successful approaches to eliminate unintended touch will need to be increasingly cognizant of the multitude of hand postures possible, the frequency of screen reorientations [Fitzmaurice et al. 1999], the likely diverse distribution of unintended touch events around the screen, and the increased frequency of intentional touch events depending on the level of detail required by the user. Although the present study did not evaluate such scenarios and we chose to eliminate touch-based feedback

Table V. Comparison of Unintended Touch Approaches

Approach	Implementation Level	Active Stylus Required?	Handedness Required?	Display Orientation / Rotation Required?	Variable Hand Postures Supported?	Bi-manual Interaction Support?
Contact Area	Firmware or Operating System	No	No	No	Yes	Yes
Hover	Operating System	Yes	No	No	Yes	No
Hand Occlusion Model	Operating System or Application	Yes	Yes	No	No	Yes
Generic screen partition (e.g., rejection regions)	Firmware	No	Yes	Yes	No	Depends on partition location
Stylus-based partition (e.g., rejection region with stylus, buffer)	Operating System or Application	Yes	Yes	Yes	No	Depends on buffer size

for the user, it is easy to see the importance of further exploring and understanding natural user behavior to inform the design of unintended touch algorithms. The abundance of pen and touch work performed over the last few years has generated many novel interaction techniques and possible gesture taxonomies, but unfortunately the most basic of knowledge is missing, for example, about the relative spacing between the hands while interacting bimanually, the orientation of the hands and stylus relative to each other and the screen, the prevalence of bimanual over unimanual interaction, and so on. Such knowledge will greatly improve the models of inking behavior that can be build and integrated within unintended touch solutions.

In what follows, the accuracy results and observed participant behavior are situated within the context of real-world demands and technological implications. Potential avenues for future research are also explored, focusing first on the benefits of different data streams and then detailing various hardware advancements and software modifications that could greatly improve the problem of unintended touch.

6.1. Factors Important for Unintended Touch

The goal of any pen and touch system or application is to allow a transfer of the seamless, interleaved bimanual interaction found in the physical world to the digital world. It is thus imperative that developers be mindful of the limitations that a given algorithm may place on interaction. In addition to accuracy, there are several other factors that need to be considered when understanding the approaches that could be viable in the future (Table V). In some contexts, such as a writing application where a user may only need to scroll periodically, a hover-based approach may work well, whereas in a sketching application, interleaved zooming and inking may be required frequently, resulting in that same hover-based algorithm to underperform and cause frustration for users.

Algorithms implemented at the application level also allow context-specific rejections and application-specific interactions, but they do so at the cost of a fragmented user experience that is inconsistent across applications and tasks. Firmware and operating systems promote a unified user experience but must be primitive, making use of a more generic, context, and functionality-free designs. Depending on the needs of a developer or the end user population, the most accurate approach may be too restrictive for the ecosystem.

Approaches that make use of handedness or harness data made available by an active stylus lead to enhanced accuracy but fragment the user experience. They are inappropriate for capacitive touchscreen devices that only support passive styli. If a lower price point is targeted, the cost of integrating an active stylus may outweigh the benefits perfect rejection would provide to users. The desire to capitalize on, and integrate, spatial data adds an additional requirement that a device must be aware of orientation, rotation, and handedness to be successful. As the detection of rotation and handedness are still in the research stage, one may be forced to sacrifice the expectations of certain populations or use cases.

Until unintended touch can be solved completely, it is imperative that the algorithms and metaphors employed are discoverable and easy to understand. When the data collection experiment began, many participants were surprised that it was technically possible to rest the skin on the screen while interacting, “It allowed me to interact so naturally—it was like I was writing on paper!” In the postexperiment debriefing, participants who had prior experience with inadequate unintended touch reaffirmed the results found by Annett et al. [2014b], expressing great frustration with current devices today: “I never know what I do that causes the screen to zoom or random marks to appear,” “It is so difficult to write with an iPad because some applications work fine, but in others my palm messes everything up and I get really upset,” and “Your system worked great! My hand never made a mark . . . I wish others were like that.” Users were very quick to determine if an approach does not work (i.e., they can’t place their hand), but they are likely to have difficulties determining why or how to adapt their behavior to overcome unintended touch. Given the fragmentation and variety of unintended touch algorithms used today, these comments highlight the need for consistent approaches to unintended touch. Designers and developers must explore a variety of use cases when designing solutions and mindfully consider ways to either alert the user if an approach falters or provide fluid, intuitive ways to overcome and avoid errors.

Given these factors and the accuracy results, it becomes clear that there is no easy solution for unintended touch. When evaluating and implementing potential solutions to unintended touch, designers and developers must prioritize and balance the requirements of their end users, the interaction they desire, and the consistency of their applications or operating systems within the tablet ecosystem and across their application suites.

6.2. Using Additional Data Streams

Although the approaches tested here used only hover and stylus location, there are many other approaches and streams of data that could be harnessed for unintended touch in the future. While the use of the hover state alone for rejection is not advocated, the “pretouch” information it provides is valuable. If decisions can begin before the user touches the screen, with either the hand or the stylus, the likelihood of incorrectly rejecting or accepting touch will decrease. Recently, some mobile phones have begun to ship with support for “touch hover,” enabling for in-air gesturing with the finger or hand [Samsung 2013; Sony 2014; STMicroelectronics 2014]. Such data could greatly improve unintended touch, as information could be obtained about the type of contact before it reaches the screen. This information could boost the performance of many of the algorithms evaluated here, in essence allowing for two-stage disambiguation and classification. Given the widespread availability of such information in the mobile world, it is only a matter of time before such information will be available on larger form factors such as tablets.

There are other stylus-based streams of data that could also be useful for unintended touch. Tablets today supporting active styli provide “pretouch” information in the form of an X,Y hover cursor. While they have information regarding the height of the stylus from the screen, this information is not yet available for use by developers. The availability of this information would provide a better picture of the time remaining before the stylus, hand, or forearm touch the screen. This information would also allow for a plethora of proximity-based interaction techniques. Integration of additional sensor information, such as pen roll or tilt, or the pressure or grip on the barrel of the pen [Hinckley et al. 2013; Song et al. 2011] could also be useful when combined with the application context. Developers, however, need to be mindful when integrating such information, as the use of additional sensors comes at an increased cost, processing time, and latency, and more complex signal processing.

When considering devices that only have a capacitive input layer, the increased popularity of axillary passive styli such as Pencil, Ink, Adnoit Jot Pro, and Pogo Connect, may become an important key to solving unintended touch on such devices [Annett 2014a]. If these styli were also able to provide information about the stylus location, similar to the Wacom Inkling [2014] or active styli, it may be possible to bring a coherent, consistent user experience to both active and passive systems.

6.3. Necessary Hardware- and Software-Based Improvements

As alluded to, the availability and frequency with which information becomes available for unintended touch will greatly influence the approaches that are possible. Today, there is a very limited time frame to collect information and make a decision regarding touch input. As system latencies decrease, it may be possible to allocate a few newly “freed” frames to unintended touch, delaying the rejection decision for each touch event by a few frames. This would allow touch input sizes to stabilize, more stylus data to become available, and temporal-based approaches to be implemented. It will, however, be a balancing act between maintaining a latency-free experience prone to false touch events versus inducing latency and eliminating unintended touch.

Modifications of firmware or operating system APIs could allow developers to receive a “rejection confidence level” along with each touch input or event. Such information could be based on a variety of factors including sensor magnitudes, duration of activity, and so on and would not introduce additional latency. There are also possibilities for systems without disambiguation as well. Visualizations could indicate the confidence a system or application has for various touch events it thinks may be unintended (e.g., those along the borders of an application or at spatially disjoint locations). Systems could vary the opacity of strokes, use colored overlays, or display specific cursors to illustrate how confident they are in the intentionality of strokes without removing them from the screen. This could help make the user aware of behaviors that cause stray markings or unintended navigations and possibly help them avoid such behaviors in the future. Visualizations could also indicate which touch points are likely from the pen and which are from the finger. Pairing these visualizations with a brushing, sweeping, or swiping gesture could help users to quickly erase these extraneous markings.

The present study required that the tablet was flat on the table and oriented toward the user in landscape mode, with only a small rotation permitted. Although rotation was corrected for in the dataset, tablets today do not have such capabilities. If tablets could report their current rotation and orientation instead of simply reporting “portrait” or “landscape,” developers could harness this information to improve the robustness of their unintended touch solutions while opening the possibility for new interaction techniques and functionalities. There are many instances where the tablet may not be flat (e.g., when held, sitting vertically, or resting on the lap [Wagner et al. 2012]), when the user could be interacting upside down (e.g., by reaching across the table to interact with someone else's device), or the tablet could be continually rotated, similar to the behavior found with artists while sketching [Fitzmaurice et al. 1999]. In these scenarios, algorithms making use of spatial information would fail, even the ones that rely on hand models. The present study did not focus on gathering information on the behavioral and postural changes that tablet rotation and orientation would produce. For this reason, it is imperative that developers carefully consider how such behaviors may influence unintended touch in these contexts.

The study also assumed that the tablet was handedness-aware and ignored the details necessary to collect this information. While some applications utilize user interface widgets and dialogs to gather such information, there has already been much work on the automatic detection of handedness [Dang et al. 2009; Ewerling et al. 2012; Ramakers et al. 2012; Wang et al. 2009; Zhang et al. 2012]. In addition to these approaches, the data collected from the digitizer suggests that it may be possible to detect handedness at the firmware level, by simply using the movement of touch blobs over time or their spatial relation to the stylus location. A firmware-based approach would eliminate the need for external cameras, augmented setups, sensors, or user interface interventions while providing valuable information for unintended touch, occlusion-free interfaces, and novel adaptive interface widgets. This exploration is left for future work.

As detailed in Section 3, companies including Atmel [2014], Samsung [SmartKeitai 2013], and Synaptics [Coldewey 2011; Sage 2011] have announced improvements in digitizer signal-to-noise ratios that have enabled better disambiguation between thin, narrow inputs (e.g., from a stylus) and larger, wider input (e.g., from fingers and palms). Once such approaches and technology exits the prototype stage and differentiated passive styli become available, it would be useful to explore more dynamic approaches to unintended touch, especially in light of the increased vocabulary of gestural commands that are becoming possible with these devices [Marquardt et al. 2011; Wigdor et al. 2011; Yan et al. 2013].

7. CONCLUSIONS

With stylus-enabled devices today, the inking experience is largely marred by the problem of unintended touch. Digitizers and applications have a difficult time distinguishing those touch events that are desired from those that are unintended. When unintended touch persists, it hampers a user's ability to interact naturally with their device and restricts the transfer of their learned behaviors from paper and pen to the digital world. This work has presented an in-depth analysis of unintended touch from both an algorithmic and a user behavior perspective.

The use of a motion capture system, along with raw data from touch and stylus digitizers allowed for an evaluation of a variety of approaches to unintended touch. In the evaluation, many novel behaviors were observed: the maximum time possible for unintended touch decisions, the typical height of the stylus during unimanual or bimanual gestural interaction, and the pattern and location of touch events. These observations were found to be important for understanding the effectiveness of various approaches and can also be useful for pen computing in general.

The analysis revealed that some current approaches, that limit the "safe" areas of the screen, such as the use of hand models or the size of touch contacts do not fare well, while others that allow for larger rejection regions such as hover or those specified by the location of the stylus, demonstrate much better performance. Deriving a general solution to solve unintended touch is difficult and influenced by factors relating to desired interaction and functionality, and information availability. Rejecting unintended touch should be made much easier in the future, however, when pretouch, handedness, device rotation, and orientation information are available for integration.

Such data will allow ample time to integrate and synthesize data before the skin has touched the surface of the screen, thus enhancing the pen and touch user experience.

REFERENCES

Tulio de Souza Alcantara, Jennifer Ferreira, and Frank Maurer. 2013. Interactive Prototyping of Tabletop and Surface Applications. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. 229–238.

Michelle Annett. 2014a. *The Fundamental Issues of Pen-Based Interaction with Tablet Devices*. Doctoral Dissertation, University of Alberta, Edmonton, Alberta, Canada.

Michelle Annett, Fraser Anderson, Anoop Gupta, and Walter F. Bischof. 2014b. The Pen Is Mightier: Understanding Stylus Behavior While Inking on Tablets. In *Proceedings of Graphics Interface*. 193–200.

Michelle Annett, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice, 2011. Medusa: A Proximity-Aware Multi-Touch Tabletop. In *Proceedings of the ACM Symposium on User Interfaces and Software Technology*. 337–346.

Michelle Annett, Albert Ng, Paul Dietz, Anoop Gupta, and Walter F. Bischof. 2014c. How Low Should We Go? Understanding the Perception of Latency While Inking. In *Proceedings of Graphics Interface.* 167–174.

Atmel. 2014. Revolutionizing the Touch User Experience. Retrieved June 2014 from http://www.atmel.com/microsite/stylus/default.aspx.

Peter Brandl, Jakob Letiner, Thomas Seifried, Michael Haller, Bernard Doray, and Paul To. 2009. Occlusion-Aware Menu Design for Digital Tabletops. In *Extended Abstracts of the SIGCHI Conference on Human Factors in Computing Systems*. 3223–3228.

William S. Buxton. 1990. A Three-State Model of Graphical Input. In *Proceedings of the IFIP TC13 3rd Interaction Conference on Human-Computer Interaction*. 449–456.

Xiang Cao, Andrew Wilson, Ravin Balakrishnan, Ken Hinckley, and Scott Hudson. 2008. ShapeTouch: Leveraging Contact Shape on Interactive Surfaces. In *Proceedings of the 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*. 129–136.

Devin Coldewey. 2011. Synaptics New Touchscreens Can Detect the Head of a Pin. Retrieved from http://techcrunch.com/2011/02/15/synaptics-new-touchscreens-can-detect-the-head-of-a-pin/.

Chi Tai Dang, Martin Straub, and Elisabeth André. 2009. Hand Distinction for Multi-Touch Tabletop Interaction. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. 101–108.

Philipp Ewerling, Alenander Kulik, and Bernd Froehlich. 2012. Finger and Hand Detection for Multi-Touch Interfaces Based on Maximally Stable Extremal Regions. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. 173–182.

George Fitzmaurice, Ravin Balakrishnan, Gordon Kurtenbach, and William S. Buxton. 1999. An Exploration into Supporting Artwork Orientation in the User Interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* 167–174.

Jens Gerken, Hans-Christian Jetter, Toni Schmidt, and Harald Reiterer. 2010. Can "Touch" Get Annoying? In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. 257–258.

Yves Guiard. 1987. Asymmetric Division of Labor in Human Skilled Bi-manual Action: The Kinematic Chain as a Model. *Journal of Motor Behavior*. 19, 4, 486–517.

Mark Hancock and Kellogg Booth. 2004. Improving Menu Placement Strategies for Pen Input. In *Proceedings of Graphics Interface*. 221–230.

Ken Hinckley and Daniel Wigdor. 2012. *Input Technologies and Techniques. The Human Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications* (3rd ed). CRC Press, Boca Raton, FL.

Ken Hinckley, Xiang Chen, and Hrvoje Benko. 2013. Motion and Context Sensing Techniques for Pen Computing. In *Proceedings of Graphics Interface*. 71–78.

Ken Hinckley, Koji Yatani, Michel Pahud, Nicole Coddington, Jenny Rodenhouse, Andy Wilson, Hrvoje Benko, and William S. Buxton. 2010a. Pen + Touch = New Tools. In *Proceedings of the ACM Symposium on User Interfaces and Software Technology.* 27–36.

Ken Hinckley, Koji Yatani, Michel Pahud, Nicole Coddington, Jenny Rodenhouse, Andy Wilson, Hrvoje Benko, and William S. Buxton 2010b. Manual Deskterity: An Exploration of Simultaneous Pen + Touch Direct Input. In *Extended Abstracts on Human Factors in Computing Systems*. 2973–2802.

Ricardo Jota, Albert Ng, Paul Dietz, and Daniel Wigdor. 2013. How Fast Is Fast Enough? A Study of the Effects of Latency in Direct-touch Pointing Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2291–2300.

Beverly Laundry. 2011. *Sheet Music Unbound: A Fluid Approach to Sheet Music Display and Annotation on A Multi-Touch Screen*. Master's thesis, University of Waikato, Hamilton, New Zealand.

Jerry Levy and Marylou Reid. 1978. Variations in Cerebral Organization as a Function of Handedness, Hand Posture in Writing, and Sex. *Journal of Experimental Psychology* 107, 2 119–144.

Chin-Lung Lin, Chia-Sheng Li, Yi-Ming Chang, Tsung-Chih Lin, Jiann-Fuh Chen, and U.-Chen Lin. 2013. Pressure Sensitive Stylus and Algorithm for Touchscreen Panel. *Journal of Display Technology* 9, 1, 17–23.

Nicolai Marquardt, Johannes Kiemer, David Ledo, Sebastian Boring, and Saul Greenberg. 2011. Designing User-, Hand-, and Handpart-Aware Tabletop Interactions with the TouchID Toolkit. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. 21–30.

Fabrice Matulic and Moria Norrie. 2012. Empirical Evaluation of Uni- and Bimodal Pen and Touch Interaction Properties on Digital Tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. 143–152.

Microsoft. 2013. Microsoft Grows Surface Family. Retrieved from http://www.microsoft.com/en-us/news/press/2013/jan13/01-22surfacefamilypr.aspx.

Microsoft. 2014. Windows Hardware Certification Requirements. Retrieved from http://msdn.microsoft.com/en-us/library/windows/hardware/dn423132.

Sundar Murugappana, Vinayak, Niklas Elmqvist, and Karthik Ramani. 2012. Extended Multitouch: Recovering Touch Posture and Differentiating Users Using a Depth Camera. In *Proceedings of the ACM Symposium on User Interfaces and Software and Technology.* 487–496.

Albert Ng, Michelle Annett, Paul Dietz, Anoop Gupta, and Walter F. Bischof. 2014. In the Blink of An Eye: Investigating Latency Perception During Stylus Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* 1103–1112.

Albert Ng, Julian Lepinski, Daniel Wigdor, Steven Sanders, and Paul Dietz. 2012. Designing for Low-Latency Direct-Touch Input. In *Proceedings of the ACM Symposium on User Interfaces and Software Technology*. 453–462.

R. C. Oldfield. 1971. The Assessment and Analysis of Handedness: The Edinburgh Inventory. *Journal of Neuropsychologia* 9, 97–113.

Openframeworks. 2009. Ofxtouch Is a Multitouch Add-On for Openframeworks. Retrieved June 2013 from https://code.google.com/p/ofxtouch/.

Raf Ramakers, Davy Vanacken, Kris Luyten, Karin Coninx, and Johannes Schöning. 2012. Carpus: A Non-Intrusive User Identification Technique for Interactive Surfaces. In *Proceedings of the ACM Symposium on User Interfaces and Software Technology*. 35–44.

Simon Sage. 2011. Synaptics Series 4 Capacitive Touchscreen Works With Stylus, Through Gloves. Retrieved from http://www.intomobile.com/2011/02/16/synaptics-series-4-clearpad/.

Samsung. 2013. How Do I Use Air Gestures? Retrieved July 2013 from http://www.samsung.com/us/support/howtoguide/N0000003/10141/120552.

Smartkeitai. 2013. Sharp. 10.1-Inch IGZO High Sensitivity Tablet Display W. Pen Input. Retrieved from http://www.youtube.com/watch?v=ofvzmluaibg.

Julie Schwartz, Robert Xiao, Jennifer Mankoff, Scott Hudson, and Chris Harrison. 2014. Probabilistic Palm Rejection Using Spatiotemporal Touch Features and Iterative Classification. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2009–2012.

SmudgeGuard. (n.d.). Eliminates Smudges and Hand Friction for Artists, Left-Handed Children & Adults, Wacom and Tablet PC Users. Retrieved June 2014 from http://smudgeguard.com.

Ke Shu. 2013. *Understanding and Rejecting Errant Touches on Multi-touch Tablets*. Master's thesis, Singapore Management University, Bras Basah, Singapore.

Itiro Siio and Hitmoi Tsujita. 2006. Mobile Interaction Using Paperweight Metaphor. In *Proceedings of the ACM Symposium on User Interfaces and Software Technologies.* 111–114.

Hyunyoung Song, Hrvoje Benko, François Gumbretière, Shahram Izadi, Xiang Cao, and Ken Hinckley. 2011. Grips and Gestures on a Multi-touch Pen. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1323–1332.

Sony 2014. Floating Touch-Developer World Mobile. Retrieved June 2014 from http://developer.sonymobile.com/knowledge-base/technologies/floating-touch/.

STMicroelectronics. 2014. Touchscreen Controllers. Retrieved June 2014 from http://www.st.com/web/en/catalog/sense_power/FM89/SC1717?s_searchtype = keyword.

Wacom. 2014. Inkling. Retrieved June 2014 from http://www.wacom.com/en/ca/creative/inkling.

Julie Wagner, Stéphane Huot, and Wendy Mackay. 2012. Bitouch and Bipad: Designing Bi-manual Interaction for Hand-Held Tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2317–2326.

Feng Wang, Xiang Cao, Xiangshi Ren, and Pourang Irani. 2009. Detecting and Leveraging Finger Orientation for Interaction with Direct-Touch Surfaces. In *Proceedings of the ACM Symposium on User Interfaces and Software Technology*. 23–32.

Daniel Wigdor, Hrvoje Benko, John Pella, Jarrod Lombardo, and Sarah Williams. 2011. Rock & Rails: extending multi-touch interactions with shape gestures to enable precise spatial manipulations. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems*. 1581–1590.

Michael Wu and Ravin Balakrishnan. 2003. Multi-Finger and Whole Hand Gestural Interaction Techniques for Multi-User Tabletop Displays. In *Proceedings of the ACM Symposium on User Interfaces and Software Technology*. 193–202.

Elba del Carmen Valderrama Bahamónde, Thomas Kubitza, Niels Henze, and Albrecht Schmidt. 2013. Analysis of Children's Handwriting on Touchscreen Phones. In *Proceedings of the International Conference on Human-Computer Interaction with Mobile Devices and Services*. 171–174.

Daniel Vogel and Ravin Balakrishnan. 2010a. Occlusion-Aware Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* 263–272.

Daniel Vogel and Ravin Balakrishnan. 2010b. Direct Pen Interaction with a Conventional Graphic User Interface. *Human Computer Interaction* 25, 4, 324–388.

Daniel Vogel, Matthew Cudmore, Géry Casiez, Ravin Balakrishnan, and Liam Keliher. 2009. Hand Occlusion with Tablet-Sized Direct Pen Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 557–566

Xiaoqi Yan, Peng Song, Chi-Wing Fu, Wooi Boon Goh, and Kwan-Liu Ma. 2013. Exploring Volume Visualizations with Whole-hand Multi-touch Gestures. In *Proceedings of Pacific Graphics*.

Dongwook Yoon, Nicholas Chen, and François Gumbretière. 2013. TextTearing for Easy Annotaiton of Digital Documents. In *Proceedings of the ACM Symposium on User Interfaces and Software Technology*. 107–112.

Robert Zeleznik, Timothy Miller, Andries van Dam, Chuanjun Li, Dana Tenneson, Christopher Maloney, Joseph LaViola Jr. 2008. Applications and Issues in Pen-Centric Computing. *MultiMedia* 15, 4, 14–21.

Hong Zhang, Xing-Dong Yang, Barrett End, Hai-Ning Liang, Pierre Boulanger, and Pourang Irani. 2012. See Me, See You: A Lightweight Method for Discriminating User Touches on Tabletop Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2327–2336.

Received March 2014; revised September 2014; accepted September 2014