Back-of-Device Interaction Allows Creating *Very* Small Touch Devices

Patrick Baudisch[1,2] and Gerry Chu[1,2]

[1] Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA

[2] Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany

patrick.baudisch@hpi.uni-potsdam.de, gerrychu@dgp.toronto.edu

ABSTRACT

In this paper, we explore how to add pointing input capabilities to *very* small screen devices. On first sight, touchscreens seem to allow for particular compactness, because they integrate input and screen into the same physical space. The opposite is true, however, because the user's fingers occlude contents and prevent precision.

We argue that the key to touch-enabling very small devices is to use touch on the device *backside*. In order to study this, we have created a 2.4" prototype device; we simulate screens smaller than that by masking the screen. We present a user study in which participants completed a pointing task successfully across display sizes when using a back-of device interface. The touchscreen-based control condition (enhanced with the *shift* technique), in contrast, failed for screen diagonals below 1 inch. We present four form factor concepts based on back-of-device interaction and provide design guidelines extracted from a second user study.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. Input devices and strategies; B 4.2 Input Output devices

Keywords: back-of-device interaction, nanoTouch, lucidTouch, mobile devices, touch, input devices, pointing.

INTRODUCTION

In this paper, we explore the question of how to provide *very small* screen devices with pointing input capabilities. Pointing input is crucial for many mobile applications from the efficient selection of links in a web page to the ability to play interactive real-time games. With *very small* we consider the following range. At the larger end, we look at devices with screens diagonals around 2.5" (6.3cm): devices designed for use during physical activities as well as tangible screen devices (e.g., *Siftables*, 2" screens [13]). At the smaller end, we look at the truly tiny screens used in accessories, such as smart watches (e.g., *Palm PDA Watch*) or electronic jewelry (e.g., [10]). The latter can be as small

as a fraction of an inch. While diminishing screen size has many impacts on overall usability, e.g. readability, in this paper we focus exclusively on input.
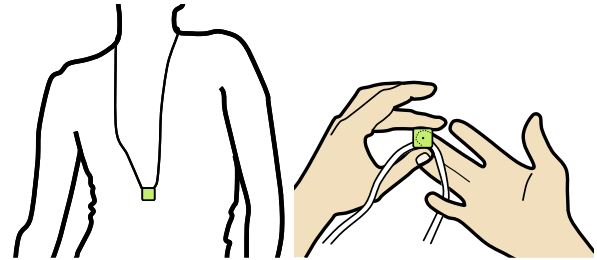

Figure 1: Back-of-device touch input can enable pointing input on very small screens. This enables building new types of devices, such as touch-capable electronic jewelry.

Recently, we have seen a departure from devices using keypads and directional-pads (Figure 2a) towards devices using touchscreens. Touch screens allow for efficient pointing input, and by eliminating the need for a keypad they allow for a comparably compact enclosure.

Intuitively, one might think that this would be a step forward towards achieving higher miniaturization (Figure 2b). Unfortunately, the opposite is true. The selection point on touch screens is ambiguous, because of the size and softness of the user's fingertip. Since the finger occludes the target area, users are required to target without visual feedback. This *fat finger problem* [21] makes the selection of small targets difficult and error-prone. Device designers address the problem most commonly by creating targets that are approximately the size of the finger. Unfortunately, this typically results in devices even larger than the more traditional d-pad devices (Figure 2c, e.g., *iPhone*).
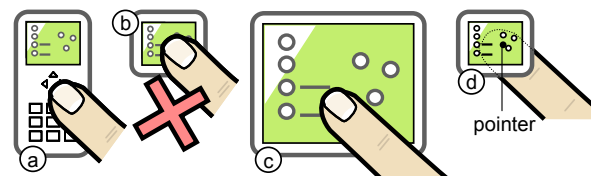

Figure 2: Pointing input on mobile devices using (a) separate d-pad, (b) touch screen of corresponding size (fat-finger problem!), (c) larger touch screen of usable size, and (d) back-of-device touch input.

Back-of-device interaction [7, 22, 27, 26] avoids interference between fingers and screen by keeping the user's finger on the back of the device, a space historically unused

(Figure 2d). An onscreen pointer informs users about the position of the finger on the back (*pseudo-transparency* [25]). Apart from that, screen contents remain visible and occlusion-free.
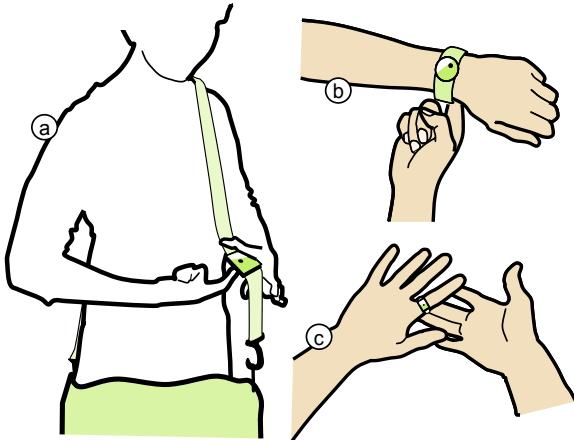

Figure 3: Three of the back-of-device designs we envision (a) *clip-on* with 2.4" screen (b) *watch* with 1.2" screen, and (b) *ring* with a screen diagonal of less than half an inch. These concepts reflect the screen sizes we used in our user study.

In this paper, we explore whether back-of-device interaction can enable pointing on *very small* touch devices. Figure 1 and Figure 3 show four devices that we are envisioning and that use back-of-device input across different degrees of smallness. These concepts inform the screen sizes we used in our user study.

Figure 4: The *nanoTouch* device offers a 2.4" screen. A dot-shaped pointer corresponds to the touch location on the back. (In the mode shown, the device facilitates discoverability by *simulating* transparency, see section "pseudo-transparency").

In order to explore this design space, we have created a prototype device we call *nanoTouch* (Figure 4). It features a 2.4" screen, like the *clip-on* design shown in Figure 3a. We simulated screen sizes below 2.4" by masking the device. Using this device, we ran a user study comparing back-of-device touch input with front-side touch input augmented with the *shift* technique [24]). We found that back-of device input continues to work for screen diagonals below 1 inch, while front-side input does not. We report the results of a second user study in which we quantify

task times and error rates of back-of-device interaction. Based on these findings, we define design guidelines.

RELATED WORK

This paper is related to targeting, touch input on small-screen devices, and back-of-device interaction.

Increasing targeting accuracy

In order to help users acquire small targets, researchers have proposed a variety of ways of enlarging targets, such as by zooming manually (e.g., double tapping [1, 18] or *rubbing* [14]) or automatically (*expanding targets* [12]). Other techniques enlarge targets in motor space only, e.g., by snapping the pointer to a target. *Starburst* [3] extends the concept to non-uniform target distributions. *Escape* combines snap-to-target with marking [28]. A similar effect can be accomplished by enlarging the pointer instead (*prince technique* [9], *bubble cursor* [6]) or by slowing the pointer down when high accuracy is required (high precision touch screen [20]).

Target ambiguity and occlusion on touch screens

Several solutions have been proposed to overcome the fat finger problem. Styli offer a precise tip and remove the user's hand from the target surface. A downside of styli is that they introduce an additional physical object that users need to retrieve before use [24].

Software techniques designed to address the fat finger problem shift finger and target away from each other. *Offset cursor* [15] creates a software pointer a fixed distance above the finger's contact point and has therefore been referred to as a software version of a stylus [24]. It uses *take-off selection* [15, 17] in which the target is selected at the point where the finger is lifted rather than where it first contacted the screen. On the flipside, the input area needs to be extended beyond the screen in order to prevent part of the screen from becoming inaccessible. On a screen measuring half an inch high, adding half an inch of space effectively doubles the device size (Figure 5a). Devices offering a touch pad on the device front are impacted in a similar way.
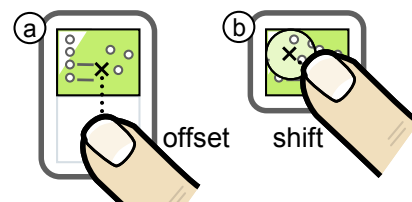

Figure 5: (a) *Offset cursor* and (b) *shift* on a device with a very small screen.

Shift [24] alleviates this limitation. Users can operate shift like a regular touch screen. If users require high precision, however, they can trigger a special precision mode by dwelling. *Shift* then "escalates" and displays a "callout" mirroring the area currently hidden under the user's finger as shown in Figure 5b. Since the callout is placed dynamically depending on the touch location, it can always be contained within the screen area and without additional

device space. Because of these benefits, we chose *shift* as the control condition in User Study 1.

Back-of-device interaction

Several researchers have proposed moving the interaction to the back of the device as means to eliminate occlusion. *BehindTouch* [7] and *BlindSight* [11] place a 12-key pad on the backside of a mobile phone. *HybridTouch* and Gummi [22, 19] allows users to scroll by performing drag gestures on a touch pad mounted on the backside of a PDA. Wobbrock enabled *EdgeWrite* on the backside of a similar device [27].

Wigdor et al. touch-enabled the bottom of their tabletop display [26]. Unlike *HybridTouch*, the touch surface in *under the table interaction* is operated in absolute mode, mapping every location on the back 1:1 to the directly opposing spot on the front.

This first generation of back-of-device interaction eliminated the occlusion of contents by the user's fingers. In exchange, however, they caused the user's fingers to be occluded by the device. While eyes-free use works fine for gestures [27], it turned out to cause large error rates when targeting ([26] requires 4.5cm targets).

LucidTouch [25] addresses the problem by introducing *pseudo-transparency*, a concept borrowed from augmented reality [2]. The device creates the illusion of a transparent device by overlaying a video image of the user's fingers (Figure 6), creating a visual effect similar to *VideoDraw* [23]. Users interact based on up to eight pointers, each of which track with one of the user's fingertips. This allows for precise manipulation independent of finger size. A more recent system, *LimpiDual Touch* [8] creates a similar effect using physical see-through.
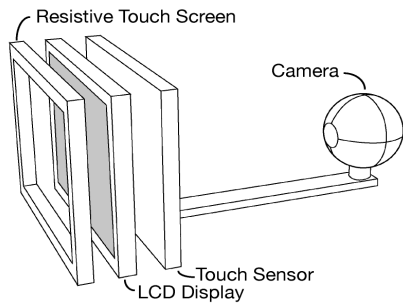

Figure 6: The LucidTouch prototype contains a 7" screen. The user's hands are tracked by a camera on a 12" boom.

THE NANOTOUCH DEVICE

Our initial plan was to use a lucidTouch device for our studies. Pilot studies, however, indicated that its size and in particular screen and bezel thickness were impacting targeting. This meant that study results obtained with this prototype would not necessarily transfer 1:1 to the space of very small devices. We therefore redesigned the device, resulting in the *nanoTouch* prototype shown in Figure 4 & Figure 7.

In order to achieve the desired size, we made several simplifications compared to lucidTouch. We decided that multi-touch would be less crucial for a device barely large enough to fit two fingers at the same time. Dropping multi-touch allowed us to trade in the camera, the boom, and the multi-touch pad for a capacitive single-touch pad. The resulting device supports the same three aspects that distinguished the original lucidTouch from its predecessors, i.e., 1:1 absolute mapping between input and screen, three states (out-of-range, tracking, and dragging), and a simplified version of pseudo-transparency.

Although we have explored some one-handed form factors, we designed the prototype primarily for bimanual use, as shown in Figure 1 and Figure 4. In this section we give a brief overview of the device, which we used to run the user studies reported in this paper.
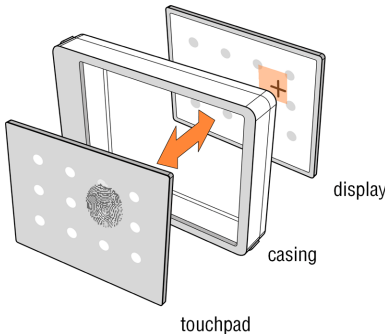

Figure 7: The nanoTouch prototype. The capacitive trackpad on the device backside is mapped 1:1 to the screen, a 320x240px, 2.4" OLED screen.

Tracking state = touch

Lacking a camera, the new device cannot sense whether a hand is hovering over the device backside. In order to still offer the required three states (*out-of-range*, *tracking*, and *dragging* [4]) we added pressure sensitivity to the pad and remapped the 3 states as shown in Figure 8: Touching the device results in *tracking*; users enter the *dragging* state by pressing.

	lucidTouch	nanoTouch
out-of-range	no hands	hovering/no hands
tracking	hovering	touching
dragging	touching	pressing

Figure 8: The three states of old and new prototype

This pressure sensing mechanism consists of a ring of elastic silicone located under the edge of the touchpad. It compresses under pressure and eventually establishes contact with a conductive path. When pressed, a solenoid inside the pad provides a brief auditory and tactile feedback confirmation, similar to the sensation produced by a micro switch. Earlier designs based on one or more micro switches had led to uneven clicking behavior.

In the default *absolute* mode, the touch pad maps 1:1 to the screen, operating like a (reversed) touch screen. Applica-

tions expecting relative (mouse) input, such as the shooter game shown in Figure 9 run the device in *relative* mode. This causes the device to function like a mirrored version of a track pad, as found in a standard notebook computer. The pressure mechanism is calibrated to require a comparably small amount of pressure. This allows for fatigue-free dragging, as required for steering activities, such as image retouching or when playing interactive video games.

Figure 9: First person shooter running on nanoTouch through the prototyping environment (Unreal Tournament 2004)

Quasi modes using thumb buttons

To allow users to trigger discrete functionality, we mounted two buttons at the bottom left corner of the device (bottom right corner for left-handed users) as shown in Figure 10. This placement allows the button to be operated using a rocking motion of the thumb of the non-dominant hand—while holding the device. Users can also *hold down* the buttons, which allows them to work as qualifier keys and thus to implement *quasi modes* [30]. We found this to work reliably and over extended periods of time. We have also used thumb buttons as a more ergonomic (even though less discoverable) alternative to pressing the touch pad and as *left* and *right* buttons when emulating a mouse. Another pair of buttons in the opposite corner of the device is available for auxiliary use.
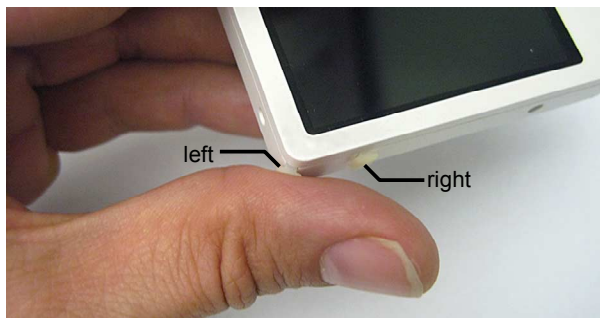

Figure 10: Thumb buttons are operated by the hand holding the device

Pseudo-transparency

One of the most useful details of lucidTouch was its naturalistic implementation of pseudo-transparency. Picking up the device with one or two hands caused an outline of these hands to show up on the screen, suggesting the screen was indeed transparent. This dramatically contributed to dis-

coverability—first-time users instantly "grasped" the concept.

To offer a similar type of experience despite the absence of the camera, the new device hallucinates an overlay based on the (x, y) coordinates of the finger received from the touch pad and a pre-rendered image of a finger. Figure 4 shows the default setting. Different users can fit the overlay to their needs by tweaking posture, finger size, and skin color.

Several details reinforce the sensation of transparency: The user's finger is rendered in front of the background, but behind the translucent buttons, suggesting a stacking order. *Pressing* is visualized by overlaying the user's fingertip with a white tip, suggesting that blood is being pressed out of the finger tip (Figure 11). The bitmap image of the finger was taken while the finger was pressed against a glass pane. A fake white reflection in the top left corner of the screen hints the existence of a glass surface in the device.
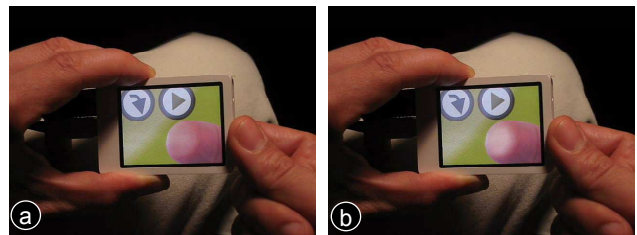

Figure 11: (a) The user's fingertip (b) turns white when pressed

In our experience, naturalistic pseudo-transparency simplifies discoverability and enables walk-up use where it would otherwise be impossible. Once users understand the concept, however, many applications will use a less naturalistic notion of pseudo-transparency—in the simplest case nothing but a dot-shaped pointer.

Development environment

Our nanoTouch prototype is tethered to a PC. This facilitates prototyping and reduces the amount of hardware in the device, allowing for a smaller form factor. The PC recognizes the device as an external screen connected via DVI. Arbitrary Windows applications can be run on the device by scaling the application window and placing it in the screen area mirrored on the device. The touchpad is connected to the PC via USB. It sends mouse move events and communicates touch and button presses as key press events.

Resulting size

The use of an OLED display (2.4", 320x240px, 166dpi) eliminates the need for a backlight and further reduces device thickness. The resulting device measures 67x47x10mm.

USER STUDY 1: BACK VS. SHIFT ON SMALL SCREENS

In order to verify our assumption that back-of-device interaction is the key to pointing input on very small devices, we conducted a controlled experiment. There were two

interface conditions: back-of-device touch interaction and front-side touch interaction enhanced with the *shift* technique. The task was to select a 1.8mm square target displayed on the built-in screen. Screen sizes varied between diagonals of 2.4" (64mm) and 0.3" (8mm). Our main hypothesis was that the back-of-device interface would work across all screen sizes, while the front-side condition would fail for screen sizes below a certain threshold. We were also interested in finding that threshold.

Screen sizes

There were four screen size conditions: 2.4", 1.2", 0.6", and 0.3" (320x240, 160x120, 80x60, and 40x30px respectively). The 2.4" condition was implemented using the full screen of a nanoTouch device. The smaller screen conditions were implemented by masking the screen in software, as shown in Figure 12 (physical masking had impacted tracking in the *shift* condition during pilots).
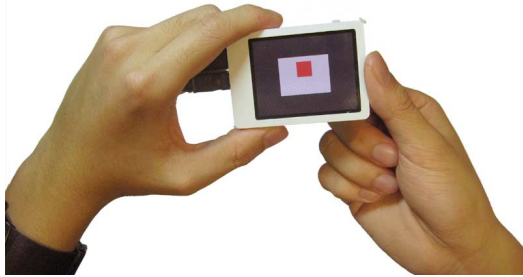

Figure 12: We simulated smaller screens, such as this 1.2" condition by masking the screen in software.

Interfaces

There were four interface conditions, all of which were implemented on a nanoTouch device.

In the two *back* conditions, participants provided pointing input via the device backside. The device was run in absolute mode. To eliminate variation due to differences in targeting strategy, participants were encouraged to keep their index finger in contact with the device at all times.

The *back* condition was broken down into two sub-conditions for the method of committing. In the *back-press* condition, participants committed selections by pressing the touch pad. The target location was determined on release. In order to minimize tracking errors, we applied a filter removing the last 80ms of move events before the "up" event, as suggested by [5]. In the *back-button* condition, participants committed selections bi-manually by pressing and releasing the corner thumb button using the non-dominant hand holding the device. This version was inspired by a study of Li et al, which suggests that mode switching with the non-dominant hand offers performance benefits [31].

In the two *shift* conditions, participants acquired targets using the *shift* technique described earlier. These conditions were implemented by overlaying a clear touch screen onto a nanoTouch device (a 3M MicroTouch 6.7" as shown in Figure 13). The device was rested against the edge of a

table to prevent the weight of the touch overlay from causing fatigue.

We ran the original *shift* code published in [24], adjusted to render at original scale on the high-dpi lucidTouch screen. The 2.4" and 1.2" screen size conditions used the original settings: a 16mm callout presented at an 11mm offset. For the two smaller screen size conditions, we reduced callout diameter and offset to fit the callout into the screen. This resulted in an 11mm callout at 8mm offset in the 0.6" condition and a 4mm callout at 4mm offset in the 0.3" condition. We also optimized the layout algorithm for the tiny screens, always placing the callout in the quadrant opposite of the initial touch location. Participants operated the shift condition using their finger tip. For the 1.8mm target size used in the study *shift* escalated instantly.
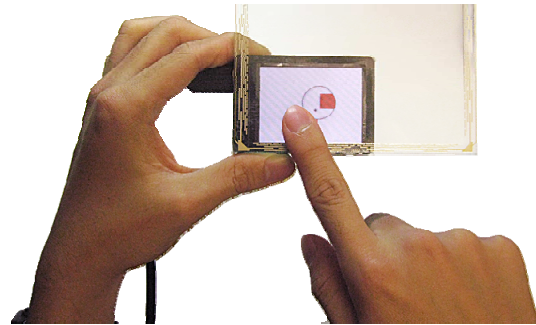

Figure 13: The *shift* conditions were run on a nanoTouch device with an overlaid touch sensor, here in the 2.4" condition.

There were two *shift* sub-conditions: when using *shift-takeoff*, users committed by lifting their finger off the screen, as described in the original paper. Improving on the published algorithm, we added the same filter used by the back condition: upon takeoff, all mouse movements occurring during the last 80ms were removed. In the *shift-button* condition participants committed using the thumb button.

Task

Participants performed a target acquisition task similar to the one described in the original shift paper [24]. Participants started each trial by tapping a start button. This removed the start button, displayed the target (red square, 12px/1.8mm), and started the timer. Now participants placed the pointer over the target and committed using press, take-off, or button press, depending on interface condition. This stopped the timer. Targets turned light red on hover (with *back*) and light blue on press (both *back* and *shift*). If a target was selected correctly, a clicking sound was played; otherwise, an error sound was played and the trial was repeated (except in the 40x30 shift condition, where high error rates made repetition impracticable).

Targets appeared in one of 9 different locations on screen, placed in the centers of the cells of a uniform 3x3 grid.

The use of the start button allowed us to control for distance between the four interface conditions on the same screen size. However, we did *not* control for distance between screen size conditions for the following reason: for

the 0.3" conditions, we had to use the entire screen as a start button (30x40px) to keep fatigue manageable. For the larger screen sizes, we could have kept distances comparable by using the same 30x40px start button and placing the target under it. The resulting task felt contrived and taught us little about targeting on the respective screen sizes. We instead chose to use longer distances as the larger screen sizes permitted them. While we again used the entire screen in the 0.6" condition, we used a 7.3mm (48px) start button for the 1.2" and 2.4" conditions, placed 9mm (60px) and 18mm (120px) from the target respectively along a vector from the target to the screen center.

Experimental design

The study design was 2 × (2 × 4 × 9) [Commit Method × (Interface × Screen Size × Target Position)] with 3 repetitions for each cell. The two types of commit methods for each interface condition was a between-subjects variable. Screen sizes were 2.4", 1.2", 0.6", and 0.3"; target positions were the 9 centroids of a regular 3 x 3 grid. For each trial, we recorded task completion time and error. Interface order and screen sizes were counterbalanced. Target positions were randomized. Half of the participants used *back-press* and *shift takeoff*; the other half used *back-button* and *shift-button*.

Participants received up-front training and at the beginning of each block. The study took about 45 minutes per participant.

Apparatus

The experiment was run on two nanoTouch devices, one of which was augmented with a touch screen for the shift condition. The devices were connected to a PC running Windows Vista, driven by an nVidia graphics card. The study was implemented in C#.

Participants

16 volunteers, (12 male) between the ages of 22 and 40 were recruited from our institution. Each received a lunch coupon for our cafeteria as a gratuity for their time. All but 3 had some experience with touchscreens. All were right-handed.

Hypotheses

We had three hypotheses: (1) we expected the shift condition to perform worse with decreasing screen size because of the occlusion problem. (2) For the smaller screen sizes, we therefore expected *shift* to have higher error than *back*, which we did not expect back to be impacted by screen size. We expected to see the same trend for task time. (3) We hypothesized that the error rate for *back-button* would be less than that for *back-press*.

Results

Error rate

Figure 14 shows participants' error rates.

We performed an analysis of variance. The model was a CommitMethod[2] x (Interface[2] x ScreenSize[4]) mixed model ANOVA. For violation of sphericity we used a Greenhouse-Geyser adjustment for degrees of freedom.

We found that there were significant main effects for both *Interface*, ($F_{1,14}$=15.8, p<0.005) and for *ScreenSize*, ($F_{1.5,20.5}$=20.9, p<0.00005). As expected, there were interactions between *Interface* × *ScreenSize* ($F_{1.8,25.8}$=42.8, p<0.0001) and between *Interface* × *ScreenSize* × *Commit Method* ($F_{1.8,25.8}$=5.1 , p<0.05).

We performed 26 post-hoc paired-sample t-tests with Bonferroni correction (α=0.0019). Error rates were aggregated across target position.
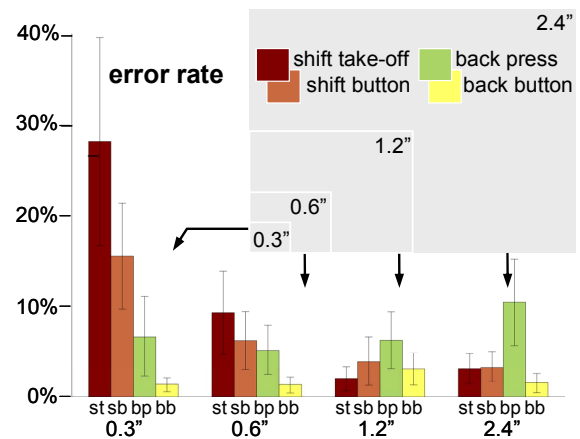

Figure 14: Error rates for the Back and Shift conditions (+/- standard error of the mean). The nested gray rectangles show the screen sizes to scale.

H1: Post-hoc t-tests for the shift condition, aggregated across *button*/*takeoff* revealed significant difference between the 0.3" condition and the other three screen sizes (all p<0.001). The differences between the 0.6" condition and the two larger conditions were borderline significant (p<0.015, p<0.006 respectively). This supports our hypothesis, that the performance of the shift condition was indeed impacted by screen size.

H2: Paired-sample t-tests between interface conditions within commit methods for the two smaller screens were either significant or borderline significant. For the 0.3" screen condition, *back-button* was less error prone than *shift-button* (p<0.00005); the difference between *take-off* and *press* was borderline significant (p<0.003). The comparisons for the 0.6" screen were both borderline significant (p<0.014, p<0.02).

This supported our hypothesis that the *back* conditions were less error prone than the corresponding *shift* conditions for small screens. As expected, we found no significant difference in error rate for *back* for six paired comparisons across screen sizes (aggregating across *button*/*press*). Also as expected, the advantage of the *back* conditions for small screens does not carry over to larger screens. The differences in error rates between *back* and *shift* for the two larger screens were not significant (p>0.05). For the 2.4" screen, *back-press* actually had a *higher* error rate compared to *shift-takeoff* (p<0.01).

H3: We performed an independent samples t-test of *button* vs. *takeoff*/*press* for *back* and *shift*, aggregating across screen size. *Back-button* was significantly less error prone than *back-press* ($p<0.00001$). There was no significant difference between *shift-button* and *shift-takeoff*.

Task time

Figure 15 shows participants' task times.

To correct for the skewing common to human response time data we based our analyses on the median response time across repetitions for each participant for each target location. We then averaged across the target locations.
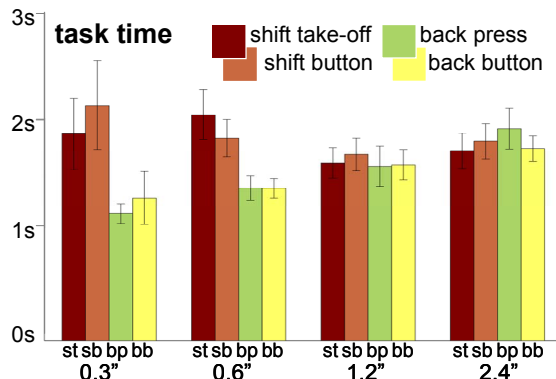

Figure 15: Task times for the Back and Shift conditions (+/- standard error of the mean)

We performed an analysis of variance. There were significant main effects for Interface ($F_{1,14}=33.9$, $p<0.00005$) and for Interface × ScreenSize ($F_{1.5, 20.3}=11.0$, $p<0.005$).

Since screen size conditions differed in distance, our data does not allow us to tell how far screen size impacted task time. Note though, that task times decrease with decreasing screen size for the *back* conditions (as explained by the shorter distances) and also roughly increase with the shift conditions. This is consistent with our observations with respect to error rates, *shift* performed poorly for very small displays.

We performed 8 post-hoc paired-sample t-tests with Bonferroni correction ($\alpha=0.00625$) to test H2 (task time for *back* would be faster than for *shift* for the same commit method and screen size). For the 0.3" screen size, the difference in speed between the two interface conditions *back-press* and *shift-takeoff* was borderline significant ($p<0.019$) as was using *button* ($p<0.031$). Using the 0.6" screen size, the difference was significant ($p<0.001$). With the two larger screen sizes, the differences were not significant.

Discussion and revisitation of the fat finger problem

As hypothesized, the back-of-device conditions outperformed the shift conditions in the very small screen conditions. This resulted from the fact that the performance of the shift conditions decreased with decreasing screen size, while the performance of the back-of-device conditions remained largely unaffected.

The decreasing performance of shift is expected, because *shift*, like any front-side touch technique can evade the fat finger/occlusion problem only to a certain extent. While the occlusion problem has traditionally been considered a problem affecting the visibility of a *target*, on very small screens the problem escalates to a more general *content occlusion problem*. Since the screen is so small with respect to the user's finger, visually communicating *anything* can become difficult as soon as the user's fingers make contact with the screen. This includes the visual presentation of targeting aids, such as *shift*'s callout.

Another perspective on the problem involves the notion of *duration*. The spatial aspects of the occlusion problem are straightforward: as illustrated by Figure 16, occlusion gets worse with (a) smaller screens, (b) larger fingers, (c) more fingers, and (d) further finger-reach across the screen. The other main factor, however, has gone unnoticed so far: the extent of the occlusion problem is in fact the product of occluded screen surface *times the duration of the occlusion*. And while front-side targeting aids can reduce occlusion, they generally do so at the expense of targeting time [24], which in turn creates additional occlusion.
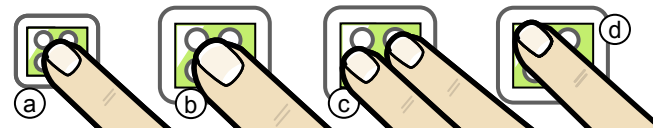

Figure 16: Factors impacting the fat finger problem

The other main finding is that back-of-device interaction allows for high accuracy across screen sizes. When triggered using the non-dominant hand (the *back-button* condition) error rates averaged 2%. This condition did substantially better than the back-press condition with error rates up to 12%. The latter value should be understood as an upper bound though; the pressure-based mechanism is clearly an early version and the error rate should be expected to shrink with improved engineering.

In summary, the study supports our hypothesis that back-of-device interaction continues to work on very small screens. It suggests that back-of-device interaction is indeed a viable approach for bringing pointing input to very small screen devices.

LAND-ON, PRECISION, AND ESCALATION

The user study presented above focused on targeting accuracy. Consequently, we have talked only about one of nanoTouch's interaction styles, namely the one that allows for high precision. That interaction style is targeting *with* visual control, as made possible by pseudo-transparency [25]. Since it allows for precise manipulation we will refer to this also as *precision* interaction (Figure 17).
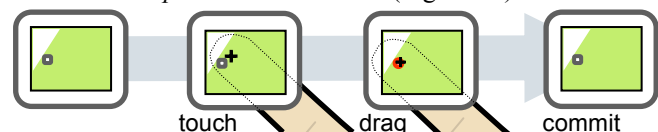

Figure 17: *Precision* acquisition of a small target

The other interaction style supported by nanoTouch is targeting *without* visual control (Figure 18), as introduced by earlier back-of-device designs, such as *under-table-interaction* [26]. We will call these *land-on* interactions. While *land-on* selection at the bottom side of an interactive table led to large error (requiring 4.5cm targets [26]) we would expect more reasonable values on nanoTouch: the prototype is small enough to allow users to see a good amount of their finger. One might hypothesize that this allows users to estimate the touch position by extrapolating their finger.
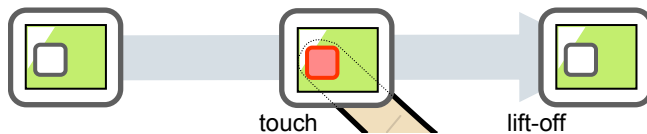

touch   lift-off

Figure 18 *Land-on* acquisition of a large target

In order to offer both interaction styles at once, we can combine precision and land-on targeting into a single interaction model (Figure 19). In such a combined model, all interactions initially proceed as a land-on interaction. However, the user can request help, e.g., by holding contact with the touch surface beyond a certain time threshold (dwelling). NanoTouch then responds by revealing the pointer, allowing users to complete the task as a *precision* interaction. Since this process resembles *shift's*, we call it *escalation*. Compared to shift's callout, however, nanoTouch's pointer is fairly unobtrusive. This makes it reasonable to escalate early and in most cases we will do so instantly. In this case, there is no more distinction on the device level, but merely a distinction between two interaction styles.
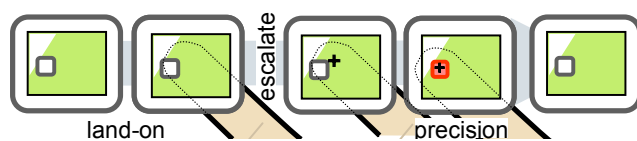

land-on   precision

Figure 19: Escalation: *land*-on to *precision*

USER STUDY 2: TARGETING ON THE BACK

So now that we know that back-of-device interaction has promise, what should an application on, say, a nanoTouch *clip-on* look like?

Before application designers can start designing controls and write applications for such a device, they need to know the constraints inherent to back-of-device interaction. For touch screens, for example, we know that users can reliably acquire targets of about 18mm and this knowledge drives the design of all higher-level components, from menus to applications. But what are the respective constraints for back-of-device interaction?

To begin answering this question we conducted a second user study. Participants in this study performed two tasks. The purpose of the *precision* task was to determine the relationship between target size and take-off task time/error

rates. The purpose of the *land-on* task was to determine land-on accuracy.

Note that the objective of this study was *not* to test a hypothesis, but make design recommendations.

Interfaces

There were three interface conditions, all of which were implemented using a nanoTouch device. Unlike the previous study, we kept screen size constant at 2.4 inches.

The *precision-press* and *precision-button* interfaces corresponded to the *back-press* and *back-button* conditions in our first user study.

In the *land-on* condition, participants acquired targets by tapping the device backside without receiving any type of visual feedback from the device.

There was no control condition, since we were only interested in determining performance metrics for nanoTouch itself, as explained above.

Tasks

There were two tasks.

The *precision* task was identical to the first user study: Participants clicked an 11mm (72px) start button to reveal the target and start the timer, and then acquired the square target. The pointer was shown immediately upon touch. Target sizes varied from 1.4mm, 2.8mm, 5.5mm, to 11mm (9, 18, 36, and 72px, respectively). We added an additional smallest target size of 0.6mm (4px) to the *button* condition; piloting had indicated that this target size was too error prone in the *press* condition. Use of the start button controlled for distance 18.4mm (120px); the start button was placed as in Study 1.

Half of the participants completed this task with the *press* interface, the other half with the *button* interface.

Participants performed the *land-on* task using the *land-on* interface. Unlike the *precision* task, we did *not* vary target size. Instead, the screen always showed a point-sized target, indicated by a crosshair. The participants' task was to tap the device backside as close to the target as possible. We recorded the relative location of the tap with respect to the target.

Experimental design

The design of the precision task was 2 x (4 x 12) [Commit method × (Target Size × Target Position)] with 5 repetitions for each cell. As mentioned above, the *back-button* condition was also tested against a 4px target, for a total of 5 targets. Target positions were the 12 centroids of a regular 4 x 3 grid. Method of committing was a between-subjects variable: half of the participants used *back-press* and the other half used *back-button*. For each trial, we recorded task completion time and error. Task order was counterbalanced. Target sizes and positions were randomized. Participants received up-front training.

The design of the land-on task was within-subjects. There were 12 target positions and 24 repetitions per position.

Target positions were the 12 centroids of a regular 4 x 3 grid. Participants received up-front training.

Apparatus
Same as in User Study 1.

Participants
14 volunteers, 11 male, between the ages of 23 and 47 were recruited from our institution. Each received a lunch coupon for our cafeteria as a gratuity for their time. All but 2 had experience with touchscreens. None of them had participated in Study 1. All were right-handed.

Hypotheses
The main purpose of the study was to determine error rates and task times for individual target sizes, positions, and interfaces in order to inform the design of back-of-device user interfaces. As a result, we had no specific hypotheses, beyond the obvious expectation that *precision button* would be less error prone than *precision press*.

Results
The findings of this study are captured in the following four charts.

Precision interaction
Figure 20 shows the error rates for *precision-press* and *precision-button*. Figure 21 shows the respective task times.
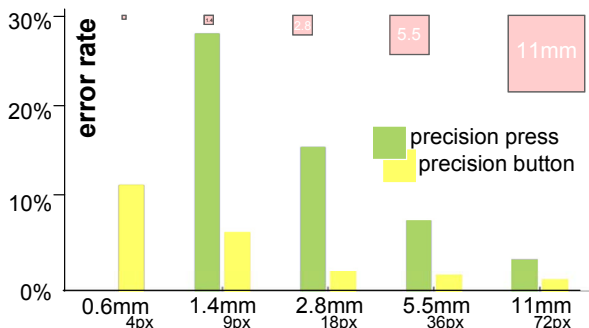

Figure 20: Error rates for the two precision conditions. Pink squares show the size of the respective targets (to scale).
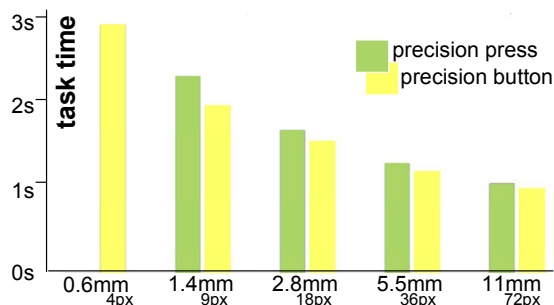

Figure 21: Task times for the two *precision* conditions.

Land-on interaction
Figure 22 shows the distribution of touch locations for all participants for the land-on task.

Figure 23 aggregates the targeting data from Figure 22. The blue line on top shows what percentage of taps was located *outside* of a square box of a given size centered around the target. This value can be used as provides an estimate for the error rate that a square button of the respective size would offer (obviously a rough estimate only, as the presence of a buttons can affect performance [32]).

The blue line ends at 80px, where the point cloud of edge targets gets clipped at the edge of the device. The magenta, middle line is based only on the two center targets, prevents it from getting cropped. The red line at the bottom is based on the same data, but assumes perfect calibration of the device, such that the centroids of the tapping data (for each user) coincide with the target.
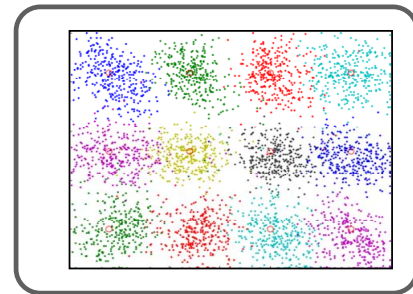

Figure 22: *Land-on:* spread of touch positions (requires color). Red rings indicate target locations (to scale).
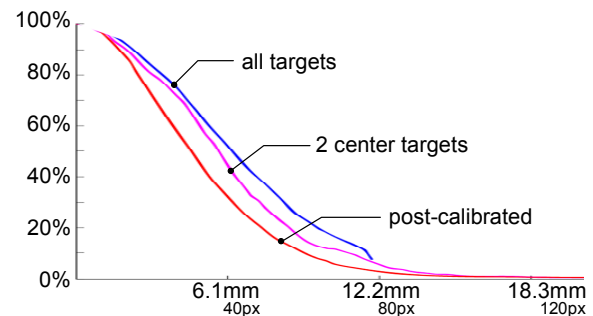

Figure 23: *Land-on:* Percentage of taps that lied outside of a square box of a given size.

Target acquisitions with land-on, required on average 884ms.

Discussion & application
The data shown in Figure 20 to Figure 23 allows us to estimate task times and error rates for acquiring different types of user interface elements.

For example, when designing a soft keyboard on the backside of a 2.4" device we could choose a 12-key design. Figure 23 predicts that, with user-specific calibration, error rates with be around 2% per keypress for the 80x80px buttons. An alternative full keyboard design might break the device backside into 8 x 6 buttons measuring 40x40px each. A land-on error rate of close to 40% suggests that we should use precision mode instead. If used with a separate button, we reach an error rate of about 2% per key press.

We can use the same process to compute estimates for task times and to compute estimates for different screen sizes, such as the 1" diagonal pendant from Figure 1.

CONCLUSIONS & FUTURE WORK

In this paper, we have argued that the key to touch-enabling very small devices is to touch-enable the device *backside*. While we have demonstrated the effect only for a specific technique—*shift*—it seems reasonable to claim that *any* pointing technique using the device front will run into the fat finger/occlusion problem once the screen gets smaller than the technique-specific threshold. The presented back-of-device design, in contrast, works practically independent of device size.

This opens up a very large space of new device designs, including the ones shown in Figure 1 and Figure 3. They allow us to take a fresh new perspective on a space where touchscreen-based designs have not been able to succeed to date, such as watch-like form factors.

In this paper, we made four contributions: (1) *nanoTouch*, a back-of device small-screen device prototype, (2) a user study showing that back-of-device interaction works independent of device size, while front-touch combined with *shift* fails for screen sizes below one inch, (3) a second study providing data that application designers can use to make UI design decisions, and (4) four back-of-device concepts ranging from in size from *ring* to *clip-on* (Figure 1 and Figure 3).

As future work, we are planning on directing our attention to *interactive* back-of-device applications, taking a closer look at steering and tracking. We also plan on exploring the visual design aspects of back-of-device applications. And finally, we plan on creating additional prototypes that explore new application scenarios.

Acknowledgements

REFERENCES

1. Albinsson, P. Zhai, S. High precision touch screen interaction. In *Proc. CHI'03,* 105-112.
2. Azuma, R.T. (1997). A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments* 6(4) (August 1997). pp. 355-385.
3. Baudisch, P., Zotov, A., Cutrell, E., and Hinckley, K. Starburst: a Target Expansion Algorithm for Non-Uniform Target Distributions. In *Proc. AVI'08*, pp. 129-137.
4. Buxton, W. A Three-State Model of Graphical Input. In *Proc. INTER-ACT '90*. pp. 449-456.
5. Buxton, W., Hill, R. & Rowley, P. Issues and techniques in touch-sensitive tablet input. *Proc. SIGGRAPH'85,* pp. 215-223.
6. Grossman, T, Balakrishnan, R. (2005). The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proc. CHI'05,* 281-290.
7. Hiraoka, S., Miyamoto, I., Tomimatsu, K. Behind Touch, a Text Input Method for Mobile Phones by The Back and Tactile Sense Interface.

8. Iwabuchi*., M.,* Kakehi. Y., and Kakehi, T. LimpiDual Touch: Interactive Limpid Display with Dual-sided Touch Sensing. In *SIGGRAPH'08* posters.
9. Kabbash, P., Buxton, W. The "Prince" technique: Fitts' law and selection using area cursor. In *Proc. of CHI'95,* 273-279.
10. Labrune, J,B, and Mackay, W. Telebeads: Social Network Mnemonics for Teenagers. In *Proc IDC '06,* pp. 57–64.
11. Li, K., Baudisch, P., and Hinckley, K. BlindSight: eyes-free access to mobile phones. In *Proc. CHI'08*, pp. 1389-1398.
12. McGuffin, M., and Balakrishnan, R. Acquisition of Expanding Targets. In *Proc. CHI'02*, pp. 57-64.
13. Merrill, D., Kalanithi, J., and Maes, P. Siftables: Towards Sensor Network User Interfaces. In *Proc. TEI'07*, pp. 75–78.
14. Olwal, A., Feiner S. (2003) Rubbing the Fisheye: precise touch-screen interaction with gestures and fisheye views. In *Conf. Companion. UIST'03,* pp. 83-84.
15. Potter, R., Weldon, L., Shneiderman, B. (1988). Improving the accuracy of touch screens: an experimental evaluation of three strategies. Proc. CHI' 88, pp. 27-32.
16. Ramos G., Boulos, M., and Balakrishnan, R. Pressure Widgets. In *Proc. CHI'04,* pp. 487–494.
17. Ren, X., Moriya, S. (2000). Improving selection performance on pen-based systems: a study of pen-based interaction for selection tasks. ACM TOCHI. 7(3):384-416.
18. Roudaut, A., Huot, S., and Lecolinet. E. TapTap and MagStick: Improving One-Handed Target Acquisition on Small Touch-screens. In *Proc. AVI'08*, pp. 146-153
19. Schwesig, C., Poupyrev, I., and Mori, E. (2004). Gummi: a bendable computer. In *Proc. CHI'04*, pp. 263-270.
20. Sears, A., Shneiderman, B. (1991). High precision touchscreens: design strategies and comparisons with a mouse. *Int. J. Man-Mach. Stud. 34*(4):593-613.
21. Siek, K.A., Rogers, Y., and Connelly, K.H. Fat Finger Worries: How Older and Younger Users Physically Interact with PDAs. In *Proc. INTERACT'05*, pp. 267–280.
22. Sugimoto, M. Hiroki, K. (2006). HybridTouch: an intuitive manipulation technique for PDAs using their front and rear surfaces. In *Proc. MobileHCI '06,* p. 137-140.
23. Tang, J. C. and Minneman, S. L. VideoDraw: a video interface for collaborative drawing. In *Proc. CHI '90*. p. 313-320.
24. Vogel, D. & Baudisch, P. Shift: A Technique for Operating Pen-Based Interfaces using Touch. *Proc. CHI'07*, pp. 657-666.
25. Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J., Shen, C. Lucid-Touch: A See-Through Mobile Device. In *Proc. UIST 2007,* pp. 269–278.
26. Wigdor, D., Leigh, D., Forlines, C., Shipman, S., Barnwell, J., Balakrishnan, R., Shen, C. Under the Table Interaction. In *Proc. UIST'06,* 259-268.
27. Wobbrock, J.O., Myers, B.A. and Aung, H.H. (2008) The performance of hand postures in front- and back-of-device interaction for mobile computing. *International Journal of Human-Computer Studies.* To appear.
28. Yatani, K., Partridge, K., Bern, M., and Newman, M. Escape: A Target Selection Technique Using Visually-cued Gestures. In *Proc. CHI 2008*, pp. 285–294.
29. Zeleznik, R., Miller, T., and Forsberg, A. Pop through Mouse Button Interactions. In *Proc. UIST'01*, pp195–196.
30. Hinckley, K., Guimbretiere, F., Baudisch, P., Sarin, R., Agrawala, M., Cutrell, E. The Springboard: Multiple Modes in One Spring-Loaded Control. In *Proc. CHI'06*, pp. 181-190.
31. Li, Y., Hinckley, K., Guan, Z., Landay, J. A. Experimental Analysis of Mode Switching Techniques in Pen-based User Interfaces. In *Proc. CHI'05*, pp. 461-470.
32. Zelaznik, H.N., Mone, S., McCabe, G.P. and Thaman, C. (1988) Role of temporal and spatial precision in determining the nature of the speed-accuracy trade-off in aimed-hand movements. *Journal of Experimental Psychology: Human Perception and Performance 14* (2), 221-230.